MASTER IN COMPUTER SCIENCE

Study of the virtual reality technology used in the visualization of business information

Darville, Christelle; Van Espen, Stéphane

Award date:
2000

FACULTÉS UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR
INSTITUT D'INFORMATIQUE

ANNÉE ACADÉMIQUE 1999-2000

Study of the Virtual Reality Technology Used in the Visualization of Business Information

Christelle Darville
Stéphane Van Espen

Thesis submitted in fulfilment of the requirement
for the degree of master in Computer Science

Supervisor : Prof. François Bodart

RUE GRANDGAGNAGE, 21 • B-5000 NAMUR (BELGIUM)

FACULTÉS UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR

INSTITUT D'INFORMATIQUE

ANNÉE ACADÉMIQUE 1999-2000

Etude de la Réalité Virtuelle Utilisée dans la Visualisation d'Information de Gestion

Christelle Darville
Stéphane Van Espen

Mémoire présenté en vue de l'obtention
du grade de Maître en Informatique

Promoteur : Prof. François Bodart

RUE GRANDGAGNAGE, 21 • B-5000 NAMUR (BELGIUM)

Abstract

The scope of our thesis is the study of the Virtual Reality technology (VR) used in the visualization of Business Information.

Business Information Visualization (BIV) is a relatively new field and has just started to gain researchers' attention. As BIV is an area in which few theoretical and practical results have been obtained, our first aim is to analyse how far Virtual Reality can be an efficient platform to deal with the challenges raised by Business Information Visualization.

Our second aim will be the attempt to determine the general design process requirements for the development of a VR BIV application. We have thought that the best way to achieve this purpose was to design our own application. We shall present the results of this work:

- the list of the most common objects composing the VR charts
- some technological requirements that the implementation platform must fulfill
- a possible architecture designed for the creation of VR BIV applications

Résumé

Notre mémoire a pour objet l'étude de la Réalité Virtuelle utilisée dans la visualisation d'Information de Gestion.

La Visualisation d'Information de Gestion (Business Information Visualization - BIV) est un domaine relativement neuf qui commence à peine à gagner l'intérêt des chercheurs. La BIV est un champs dans lequel peu de résultats théoriques ou pratiques ont été obtenus. Par conséquent, notre première intention est d'analyser dans quelle mesure la Réalité Virtuelle fournit une plateforme capable de relever les défis soulevés par la BIV.

Par ailleurs, nous tenterons de déterminer les prérequis généraux du processus de conception de développement d'une application utilisant la Réalité Virtuelle à des fin de Visualisation d'Information de Gestion (VR BIV application). Afin d'atteindre ce but, nous avons créé notre propre application. Nous présenterons ainsi le fruit de notre travail:

- la liste des objets les plus couramment utilisés dans la composition de VR Charts
- quelques prérequis technologiques devant être remplis par la plateforme d'implémentation
- enfin, la proposition d'une architecture conçue en vue de créer des "VR BIV applications".

Acknowledgements

Nos remerciements les plus sincčres s'adressent

Au Professeur François Bodart, pour son soutien permanent et son suivi rigoureux, de l'ébauche à la clôture du mémoire. Nous voudrions particulièrement le remercier pour les nombreux conseils et critiques dont il nous a gratifié.

To Professor Janet L. Wesson, from the University of Port ELizabeth, for her guidance and advice during our training at UPE. Also for the time she gave us during these four months. We would like to thank her, as well as her family and friends, for their warm welcome in South Africa.

Au Professeur Monique Noirhomme, pour l'ensemble des documents qu'elle nous a gracieusement fournis.

To Leon Nicholls, Senior Lecturer at the University of Port Elizabeth, for his precious help and his attention towards us during our whole training.

To Mr Dekock, Director of the University of Port Elizabeth, and Mrs Dekock, for their kind reception at UPE.

To the whole staff of UPE, professors and secretaries, for their kindness and warm receptions.

To Ph.D. Ping Zhang, from the Syracus University (New-York), for all the precious documents she sent to us cordially.

A Thibaut Bodart, pour ses critiques et précieux conseils, pour tout ce temps passé à vérifier notre travail.

A M. Meurice, pour le temps consacré à la correction de notre anglais et à ses conseils avisés concernant l'usage de la langue de Shakespeare.

A tous les étudiants de troisième maîtrise pour tous les moments passés ensemble.

Aux membres de nos familles pour les nombreux échanges de courrier électronique lors de notre stage à Port Elizabeth et pour les encouragements qu'ils nous ont adressés depuis lors.

A Christophe "Tchou" Darville, sans l'aide de qui ce mémoire ne serait pas si coloré.

Finalement, et le plus important pour nous, à nos parents, sans qui nous ne serions simplement pas là aujourd'hui. Pour leur soutien depuis le premier jour. Pour nous avoir permis d'arriver là où nous en sommes. En un mot, pour tout.

Contents

List of Figures

Introduction

The scope of our thesis is the study of the Virtual Reality (VR) technology used in the visualization of Business Information (BIV).

The *Virtual Reality* defines an artificial environment created with computer hardware and software and presented to the user in such a way that it appears and feels like a real environment [Lycos00b]. In order to enjoy the Virtual Reality, a wide range of solutions is available, with various quality implementations providing - at a corresponding cost - different degrees of realism and immersion. We shall focus on a simple solution based on very traditional devices (such as a screen, a keyboard and a mouse).
Virtual Reality will be discussed in chapter 1.

According to PING ZHANG [Zhang99b], assistant professor at Syracuse University,

> "Business Information Visualization is a process of creating appropriate computer-generated visual representations of large amount of non-geometric managerial data for human problem-solving and decision-making support".

Business Information Visualization is a relatively new field and has just started to gain researchers' and practitioners' attention. As BIV is an area in which few theoretical and practical results have been obtained, our first aim is not to construct a visualization theory but to analyse how far VR can be an efficient platform to deal with the challenges raised by Business Information Visualization.
From now on, the term "VR BIV" will refer to the use of Virtual Reality in the Business Information Visualization field.

Our second aim will be the attempt to determine the general design process requirements for the development of a VR BIV application. We have thought that the best way to achieve this purpose was to design our own application. We shall present the results of this work:

- the list of the most common objects composing the VR charts
- some technological requirements that the implementation platform must fulfill
- a possible architecture designed for the creation of VR BIV applications

This thesis is subdivided into two parts, each one covering an aim. The first one focuses on the analysis of the Virtual Reality usefulness in Business Information Visualization field. The second part focuses on our researches on the design process requirements for the development of a VR BIV application.

Part 1
The first part is made up of four chapters.

The first one will briefly present the different fields using the Virtual Reality technology. The following chapter will then deal with the set of problems coming with decision-making as well as the various tools designed to solve them. An introduction to semantic properties of business data and to the most common VR charts will be discussed in chapter 3. Finally, chapter 4, the core of the first part, will conclude with a discussion about assets and shortcomings of using the Virtual Reality for the Business Information Visualization field.

Part 2
The second part will focus on some general design process requirements for the development of a VR BIV application. Chapter 5 will review the list of the most common objects composing the VR charts. With this in view, description of geometric primitives and additional components constituting VR Charts will be overviewed. The next chapter will be about some technological requirements that the implementation platform must fulfill. Finally, chapter 7 will propose a possible architecture designed for the creation of VR BIV applications and chapter 8 will show an example of a VR BIV application of our own.

Part I

Analysis of the Virtual Reality Usefulness in the Business Information Visualization Field

Chapter 1

The Fields of Virtual Reality

1.1 Introduction

The term 'Virtual Reality' (VR) has been defined in a number of ways since its creation, the focus being broadened in recent years. Unfortunately, this broadening has tended to cause some confusion about what constitutes VR and what does not. What we mean by 'Virtual Reality' is a computer interface characterized by high-degrees of immersion and interaction, aiming to make the user believes, as much as possible, that he is actually inside the virtual environment [Bell95]. To 'enter' a virtual world, a user wears special gloves, head-mounted displays (earphones and goggles), all of which receiving their input from the computer system. In this way, at least three of the five senses are controlled by the computer.
Nowadays, this type of Virtual Reality solution requires extremely expensive hardware and software and are confined mostly to research laboratories [Lycos00b].

Nevertheless the term 'Virtual Reality' is sometimes used more generally to refer to any virtual world represented in a computer. With this in view, three-dimensional space that appears **on display screen** can also be assimilated to a Virtual Reality solution. The user can figuratively move within this space. As he presses keys to turn left, right, up or down, or go forwards or backwards, the images on the screen will change to give the impression that the user is moving through a real space [Lycos00a]. Throughout this thesis, we shall refer to **this definition** of the Virtual Reality technology.

The ability of Virtual Reality to simulate three-dimensional objects inside existent - or imaginary - environments has proven to be useful to education, advertising, business, military, science and entertainment industry. Virtual Reality allows to simulate *realistic* representations as well as *abstract* representations.

We shall now introduce a non-exhaustive overview of some application areas of these representations.

Note that in this chapter we shall use interchangeably both *'3D'* and *'VR'* terms.

1.2 Realistic fields of applications

Realistic representations are often used for scientific visualization as well as educational means or entertainment purpose.

Scientific visualization evolved in order to meet the ever-increasing need to deal with highly active, very dense data sources, which, for example, included satellite data, geophysical data, as well as data from supercomputer computations. It means using computer-generated graphics to help people understand and clarify visually the relationships inherent in data.

Scientific visualization emerged in the late eighties as a key field in computer science and in numerous other application domains such as geoscience, meteorology, medicine, etc. Scientific visualization provides processes for steering the data set and seeing the unseen, thereby enriching existing scientific methods. Most scientific visualization systems are natural representations of real world objects that possess **known geometric** structures [Zhang99b].

Besides, realistic representations span a broad variety of applications: educational means, interactive and multimedia titles, etc.

We propose to get a general view of four realistic fields of application: medical imaging, 3D landscape modelling, 3D as an educational means and facial animation.

1.2.1 Medical imaging

Multidisciplinary team of computer scientists, engineers, and physicians are developing solutions to *medical* problems. For instance, a medical virtual reality software system, called FreeFlight[WakeForest00], allows a user to simply import radiologic data and visualize human anatomy in an intuitive manner. FreeFlight's clinical applications are medical education and investigational protocols, including for example virtual bronchoscopy and virtual colonoscopy.

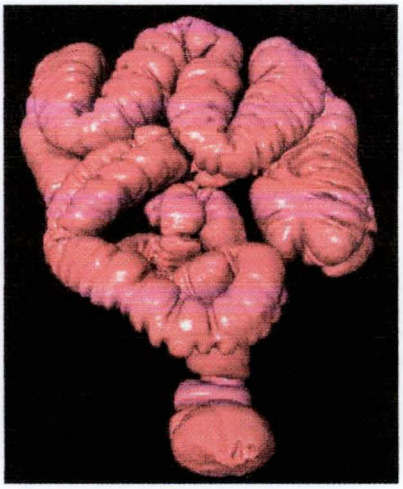


Figure 1.1: A whole colon

Moreover the software system seems to have potential for **non invasive** diagnosis and surgical guidance. On the WFUSM Virtual Endoscopy Center website[1], we can find "magnificent" 3D models of colon[2]. Such models can be generated by magnetic resonance.

1.2.2 3D landscape modelling

Since the topographic surface has a great influence on most environmental processes and many human activities, *3D landscape modelling* is an elementary field of study for spatial models.

The task of scientific teams is to build on accuracy, reliability and efficiency of digital terrain representations on both a geometric and semantic level, as well as to develop novel and improved analysis and manipulation methods. For instance, data supplied by this kind of visualization are useful for natural resource management. 3D landscape modelling is aiming at landscape architecture, architecture, forestry, and engineering.

1.2.3 3D : an educational means

Many studies have shown that students learn best when a variety of teaching methods are used, and that different students respond best to different methods. To this end, computers are being used more and more as teaching tools, to provide students with a wider variety of learning experiences [Bell95].

[1] http://www.vec.bgsm.edu/gallery/Virtual Endoscopy Center Gallery.htm
[2] Figure 1.1 displays an example of such colon model.

Figure 1.2: A 3D landscape

3D as *an educational means* uses the computer to achieve things that no book could possibly accomplish. The Virtual Forest educational program, produced by EcoLogic [EcoLogic00], displays views from inside a central Hardwoods forest in three dimensions, and lets the student rotate the view, look up or down and spin around 360 degrees as well.

Figure 1.3: A virtual forest

Students can also move a branch of a tree in any direction to see all of the features and zoom in on tiny lateral buds. 3D, as an educational means, is an efficient way to give a feel for what this forest type looks like.

1.2.4 Facial animation

Facial animation is now attracting more attention than ever before in its 25 years as an identifiable area of computer graphics.

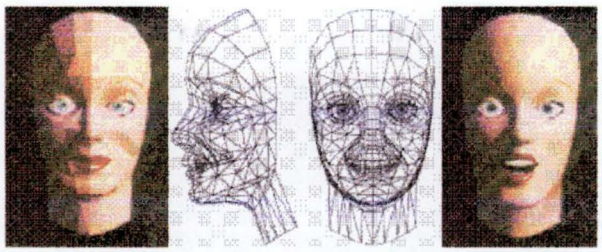


Figure 1.4: Facial animation

Applications synthesizing realistic faces - or imaginary ones - are found in sophisticated human-computer interfaces, interactive games, multimedia titles, VR telepresence experiences, and in a broad variety of production animations.

1.3 Abstract fields of applications

The *abstract* fields we propose to review are *Human-Computer Interface (HCI)*, *statistics* and *Business Information Visualization*.

1.3.1 Human-Computer Interface (HCI)

According to Microsoft [Microsoft00], the leitmotiv of *HCI* is

> "The less people have to think about how to work their computer, the more mental energy they have left for their real work."

Some research teams think that 3D will help to achieve this goal. Among others, Microsoft is developing the "Task Gallery" , a running 3D research prototype user interface that expands the desktop into an entire office with an unlimited number of desktops. The screen becomes a long gallery with paintings on the walls that represent different tasks, and the user moves from the one to the other with a series of mouse and keyboard commands.

1.3.2 Statistics

Statistics are an important part of many domains. For instance, statistics allow people in an organization to monitor different activities as well as their performance. Three-dimensional visualization is an excellent tool making interpretations of statistical results much easier.

Figure 1.5: The Microsoft Task Gallery

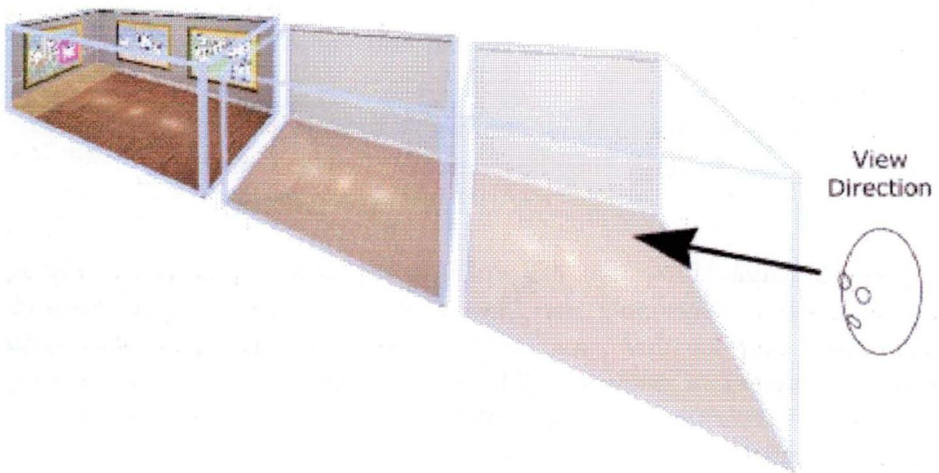


Figure 1.6: The three-dimensional desktop

For instance, the well-known histogram is used when people need to understand what a distribution of data looks like whereas scatter charts help determine the relationship between two different variables.

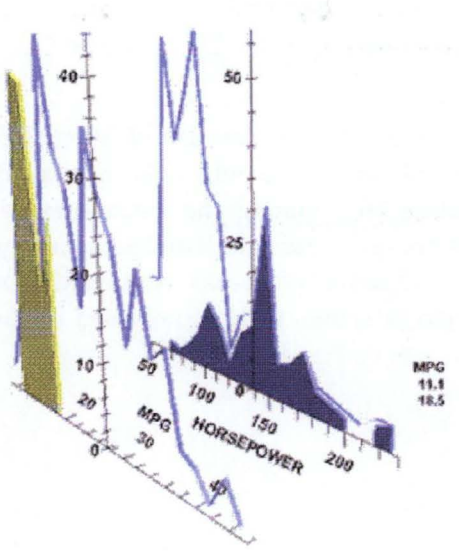


Figure 1.7: A histogram correlation

In business domains, statistical visualization tools are a powerful way to provide the ability to correlate multiple business inputs in order to identify trends, make comparisons and find connections hidden within data. For example, the histogram correlation (see picture 1.7) allows to examine multiple sets of data in order to highlight relationships and patterns existing among the data.

1.3.3 Business Information Visualization

Lately, *Business Information Visualization* (BIV) has been using Virtual Reality as a new platform.

Business Information Visualization goes about visual representations of large amount of *geometric* as well as *non-geometric* managerial data for human problem-solving and decision-making support. For instance, representation of financial information is a usual concern. The challenging aspect about BIV is trying to visualize abstract information i.e. information that doesn't have an inherently physical manifestation[3].

GHERSON and EICK explain in [Zhang99b]:

> "Information Visualization is a process of transforming data and information that are not inherently spatial, into a visual form allowing the user to observe and understand the information. This

[3]at the opposite of braim tumor that has an inherently physical manifestation

> is in contrast with scientific visualization, which frequently focuses on spatial data generated by scientific processes"

Since this is the topic of our thesis, the use of Virtual Reality for Business Visualization purposes will be developed in the next chapters. With this in view, chapter 2 will place BIV among the various existing decision-making tools. Concepts such as non-geometric data and semantic properties of business data will be developed in chapter 3 whereas assets and shortcomings of VR for the business visualization purposes will be discussed in chapter 4.

Chapter 2

Tools for Business Data Management

2.1 Introduction

The past two decades have seen a dramatic increase in the amount of information or data being stored in electronic format. This accumulation of data has taken place at an explosive rate. It has been estimated that the amount of information in the world doubles every 20 months while the size and number of databases are increasing even faster [UnivBelfast99].

It is an indisputable fact that information is at the heart of business operations and that decision-makers make use of the data stored to gain valuable insights into the business. In most management domains, business users are faced with

- an increasingly huge volume of complexed data (multi-dimensional, heteregeneous,etc.)
- multiple complex relationships among them
- the negotiability of constraints that make problem-solving overwhelming

Significant patterns of customer behaviour are buried in mountains of raw data that all look the same. As "time means money", it is necessary to use techniques capable of treating this kind of data in a reasonable length of time as well as reducing their amount without losing information.

Some techniques such as *Data Mining, OLAP*, and *BIV* are working together to provide managers with the information they need in order to make effective decisions about the strategic directions of an organization. They satisfy diverse business requirements for tracking, monitoring, and acting on key strategic, tactical, and operational business indicators.

These techniques span a variety of organizational functions [OLAP97]:

- For budgeting, activity-based costing (allocations) in finance departments,
- For financial performance analysis and financial modelling,
- For sales analysis and forecasting in sales departments,
- For market research analysis, sales forecasting in marketing departments,
- For promotions analysis, customer analysis, and market/customer segmentation,
- For production planning and defect analysis in manufacturing departments.

Understanding what is important and what is not is the challenge. Decisions makers require accurate, relevant information in order to make the best informed decisions. Short descriptions of these different techniques are now overviewed.

2.2 Data Mining

At the early stage of management, Database Management Systems (DBMS) gave only access to the data stored. This was only a small part of what could be gained from the data. Traditional On-Line Transaction Processing systems (OLTPs) are good at putting data into databases quickly, safely and efficiently but are not good at delivering meaningful analysis in return. Analysing data can provide further knowledge about a business, by going beyond the data explicitly stored in order to derive business knowledge. This is where *Data Mining* or Knowledge Discovery in Databases (KDD) has obvious benefits for any organization.

The term "Data Mining" has been stretched beyond its limits to apply to any form of data analysis. Some of the numerous definitions of Data Mining, or Knowledge Discovery in Databases are :

> Data Mining, or Knowledge Discovery in Databases (KDD) as it is also known, is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data. This encompasses a number of different technical approaches, such as clustering, data summarization, learning classification rules, finding dependency networks, analysing changes, and detecting anomalies.

WILLIAM J FRAWLEY, GREGORY PIATETSKY-SHAPIRO AND CHRISTOPHER J MATHEUS [UnivBelfast99]

> Data Mining is the search for relationships and global patterns that exist in large databases but are 'hidden' among the vast amount of data, such as a relationship between patient data and their medical diagnosis. These relationships represent valuable knowledge about the database and the objects in the database and, if the database is a faithful mirror, of the real world registered by the database.

MARCEL HOLSHEMIER & ARNO SIEBES (1994) [UnivBelfast99]

The analogy with the mining process is described as :

> Data Mining refers to "using a variety of techniques to identify nuggets of information or decision-making knowledge in bodies of data, and extracting these in such a way that they can be put to use in the areas such as decision support, prediction, forecasting and estimation. The data is often voluminous, but as it stands of low value as no direct use can be made of it; it is the hidden information in the data that is useful"

CLEMENTINE User Guide, a Data Mining toolkit [UnivBelfast99]

Basically Data Mining is concerned with the analysis of data and the use of software techniques for finding patterns and regularities in sets of data. It is the computer which is responsible for finding the patterns by identifying the underlying rules and features in the data. The idea is that it is possible to strike gold in unexpected places as the Data Mining software extracts patterns not previously discernable or so obvious that no-one has noticed them before.

The best techniques are those developed with an orientation towards large volumes of data, making use of as much of the collected data as possible to reach reliable conclusions and decisions.

2.3 OLAP

In contrast to a Data Warehouse, which is usually based on relational technology, *OLAP* (On-Line Analytical Processing) uses a multidimensional view of aggregate data to provide quick access to strategic information for further analysis. OLAP enables analysts, managers, and executives to gain insight into

data through fast, consistent, interactive access to a wide variety of possible views of information. While OLAP systems have the ability to answer "who?" and "what?" questions, it is their ability to answer "what if?" and "why?" that sets them apart from Data Warehouses. OLAP enables decision-making about future actions.

A typical OLAP calculation is more complex than simply summing data. For example : "What would be the effect on chips costs to distributors if potatoes prices went up by 10 BEF/Kg and transportation costs went down by 5 BEF/Km ?"

OLAP and Data Warehouses are complementary. A Data Warehouse stores and manages data. OLAP transforms Data Warehouse data into strategic information. OLAP ranges from basic navigation and browsing (often known as "slice and dice"), to calculations, and more serious analyses such as time series and complex modelling. Although OLAP applications are found in widely divergent functional areas, they all require the following key features [OLAP97]:

1. Multidimensional views of data[1]
2. Calculation-intensive capabilities[2]
3. Time intelligence[3]

2.4 Business Information Visualization

OLAP, entreprise reporting and Data Mining solutions prove to be often too complex, particularly when applied to time and detail critical applications, such as target marketing, customer loyalty, and product and category analysis [Hamilton00]. Demand for much higher dimensional complexity and data granularity, as well as requirements for the assimilation of multiple data sources, simultaneoulsy and at Internet speed, require too much IT intervention and can consequently lead to report explosion. This is **in direct conflict** with the needs of front line decision makers and knowledge workers **who are demanding to be included in the analytical process** and who are often placed way out of it because of intensive IT processes [Hamilton00]. According to us, *Business Information Visualization* provides an appropriate solution.

[1] i.e. rotation to new dimensional comparisons in the viewing area

[2] in order to solve complex queries that involve retrieving multiples numbers and aggregating them

[3] For instance, trend analysis over sequential time periods

Business Information Visualization is the **presentation** layer that can invoke other decision support services such as OLAP and datamining, giving the end users a single entry point into a vast array of information while the complex underlying technologies remain transparent.
BIV is suitable because most of what humans learn come through their sight [Zhang-Whinston95]. Simplifying the display of information increases the understanding of complex multidimensional relationships. Finding valuable insights and strategic outliers in data by relying on visual perception is significantly easier than filtering through multiple formatted reports and cross tab pages. BIV takes full advantage of stored database information by presenting it in a meaningful and interactive form.

Business Information Visualization focuses basically three distinct purposes [Wright98]: *discovery*, *fast cognition* and *communication*.

Discovery
Discovery helps to understand and conceptualize complex issues. Visualization tools are a very useful method of discovering patterns in data sets. They may be used at the beginning of an analytical process in order to get a rough feeling of the informational content of the data sets in which patterns are to be found. BIV is efficient in the discovery of relationships, trends, distributions, outliers, anomalies, etc. For instance, a trend in data might not be highlighted from a **table** while a **histogram** could display this trend distinctly.

Fast cognition
Fast cognition is required for real-time decision making. BIV can help decision-makers to be more effective by providing the visualization drives toward finding an answer.
Decision-makers change tasks frequently during the day, and a big chunk of time for thoughtful analysis does not exist. Fast cognition is used in order to make fast decisions and not really to gain a better understanding.

Communication
Communication purpose could be summarized like this: "Don't tell me, show me !". A single illustration can communicate a message more quickly and effectively than several pages of text. In comparison with 'Row and Column' static reports, visual display of information improves the comprehension of complex business relationships, therefore making the communication of the results easier.

We are now aware that the ultimate purpose of visualizing management data is to support human problems solving or decision-making processes. But obstacles stand up. According to [Zhang-Whinston95],

> "although computers become more and more available, the primary challenge of the computing society in the current and coming decades is not the computational power. It is collected or generated data, especially their representations to users in a comprehensible form, that affect and limit the basic capabilities of computing. It is extremely necessary to improve the communication between users and computers, to transform the vast computing-related data into some representations that help humans to understand data."

As a matter of fact, in business domains, relationships among data sets can be either *geometric* in structure[4] or *non-geometric*. Non-geometric structure means that **no obvious physical model can be used to represent the data that humans can understand objectively**. This nature of data causes a challenge for graphically representing data , i.e. concrete, comprehensible geometric structures have to be created for representing these non-geometric data. This involves a major concern : the efficiency of the final visual representations for the human vision. When creating geometrical images, it is important to consider human visual perception characteristics[5], as well as cognitive process of visual information processing. Such investigation field is focused on by [Zhang99b].

2.5 Example : a spreadsheet and its visualization

GEONAME	FEMPOPHIS	FEMPOPCUR	FEMPOPPRO	MALEPOPHIS	MALEPOPCUR	MALEPOPPRO
Alabama	2104727	2194735	2313044	1935860	2027197	2142473
Alaska	260230	289206	323009	289813	321144	356674
Arizona	1854696	2020893	2246862	1810532	1979505	2205997
Arkansas	1217920	1262166	1324423	1132805	1179480	1242514
California	14863538	15749642	16767433	14896483	15796959	16807880
Colorado	1663315	1829591	2053524	1631079	1800994	2026381
Connecticut	1694467	1685708	1677174	1592649	1589487	1584549
Delaware	343224	364186	389614	322944	343678	368088
District Of Columbia	324005	304640	282896	282895	266952	247855
Florida	6676516	7134287	7682227	6261410	6715454	7251299
Georgia	3334160	3606091	3957412	3144056	3414293	3756211
Hawaii	544330	585344	633386	563899	601348	645333
Idaho	505853	562163	636315	500896	558516	633963
Illinois	5879080	6037679	6242901	5551522	5723221	5934747
Indiana	2856315	2957899	3089520	2687844	2795029	2928395
Iowa	1432397	1453192	1480271	1344358	1369856	1399807
Kansas	1263216	1294143	1333637	1214358	1249602	1292001
Kentucky	1900357	1963079	2045975	1784939	1851043	1935280
Louisiana	2188796	2232628	2285361	2031177	2080567	2134607

Figure 2.1: A spreadsheet

[4] such as hierarchies, matrix or networks

[5] for instance, the perspective view

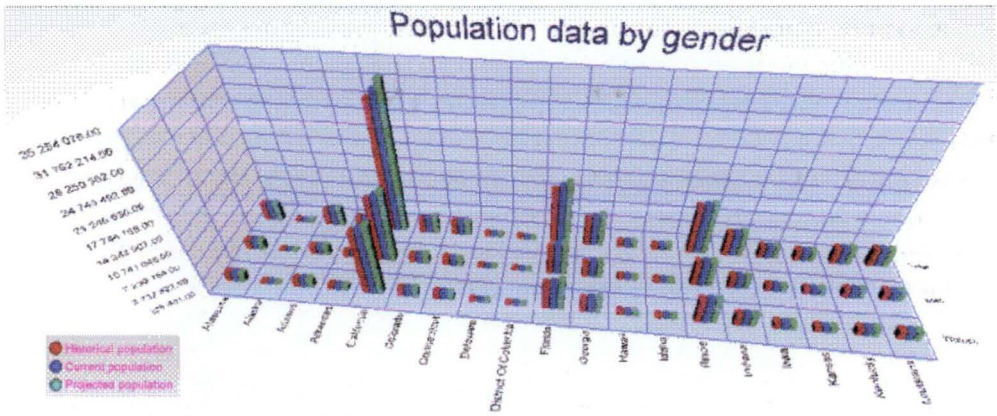


Figure 2.2: Three-dimensional visualization of the spreadsheet

The data displayed in table 2.1 are related to populations of male and female groups from some American states. A Column Chart was chosen to represent the historical, current and projected populations of men and women in the listed states.

This Column Chart was generated by a MS Excel macro supplied with the Portal i3D GraphFactory developed by EM7[6] [EM700]. The last row[7] represents the sum of the men and women rows.

If we have a look on the Column Chart only for a few seconds, we can make instantaneous conclusions:

- California, Florida, Georgia, Illinois and Indiana get the highest population, California being by far the highest.
- Population of other states are rather low.

Cross tab pages are not efficient for depicting the general nature of data, and as a consequence, managers would need more time to draw the same statements in an equivalent length of time. This simple example proves the fast cognition and the communication skills of Business Information Visualization. Besides, managers examining this chart in a virtual environment could easily obtain detailed values for each population groups and could walk around to focus on specific states.

[6] http://www.em7.com/

[7] it was automatically calculated by the macro

2.6 Conclusion

The first chapter has overviewed some significant fields in which Virtual Reality provides an appreciable support. This second chapter just explained the crucial role of Business Information Visualization in challenges raised by management and decision-making processes. It is now time to investigate what Virtual Reality can bring to BIV, and consequently, to the decision-makers. This is what the two following chapters are covering.

Chapter 3

Introduction to Semantic Properties of Business Data

3.1 Introduction

The purposes of this chapter are to introduce semantic properties of business data in the perspective of their visualization, and to introduce the most common VR Charts.

One of the interests of these purposes is to help the business users with the selection of the right charts. As a matter of fact, according to BARBARA MIREL [Mirel99], identifying the right chart for a question requires a good deal of experience. However users want to have this skill almost as soon as they start using visualizations. This competence develops late in a novice's learning curve. Nevertheless, without this competence, users are unable to make directly relevant analysis. Users of BARBARA MIREL's studies reported frustration that they were not developing it more readily.

To solve the problem raised by Barbara Mirel, we have found a rule to determine how to select the right chart: the selection of the right chart is function of the semantic properties met by each chart.

3.2 Clarifications and agreements

In order to achieve both purposes[1] correctly, we have to lay some clarifications and agreements about the main terms and notions in use further in our thesis:

[1] introduction to semantic properties of business data and introduction to the most common VR Charts.

1. "Chart" and "Graph" terms
2. Two-axes 3D vs three-axes 3D
3. 3D visualization vs Virtual Reality
4. Right-handed system
5. Data dimension vs chart dimension

3.2.1 "Chart" and "Graph" terms

First of all, it is important to note that we shall use two close terms further in this paper : *chart* and *graph*. In order to understand those two concepts correctly, we need to provide a short explanation.

A *chart* is sometimes referred to as an information graphic, a vehicle to portray data in a visual form for analysis, planning, communicating and other purposes. Charts are divided into categories including graphs, tables, diagrams and maps. *Graph* is a more specific term. It defines a chart that graphically displays quantitative relationships between two or more groups of information. *Graphs* have combinations of several axes using one or more quantitative scales. This definition differenciates *graphs* from other charts such as tables, diagrams or maps [Harris99].

Since graphs constitute one of the major categories of charts, graphs are frequently refered to as charts. Therefore, throughout this thesis, we shall use both terms interchangeably.

3.2.2 Two-axes 3D vs three-axes 3D

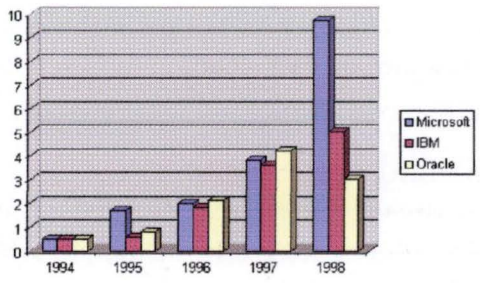


Figure 3.1: Two-axes 3D

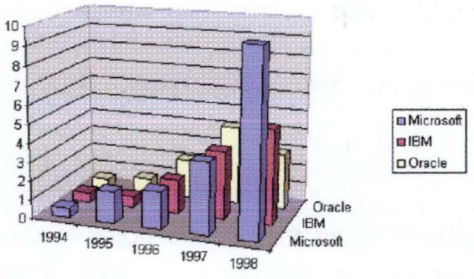


Figure 3.2: Three-axes 3D

Secondly, a clarification is required. The phrase "three-dimensional (3D) graph" is often used in two significantly different ways, which sometimes cause confusion [Harris99]. In one case the phrase is used to mean that a 2D chart has the appearance of having depth which makes it look three-dimensional. The only difference with traditional 2D charts is the cosmetic effect (see figure 3.1). In the other case, the phrase three-dimensional graph refers to a graph having three axes (see figure 3.2).

In this thesis, we only talk about three-axes charts, that we shall refer as 3D charts and VR Charts.

3.2.3 3D visualization vs Virtual Reality

We shall make a distinction between 3D visualization and Virtual Reality.

3D visualization is a visualization by means of static 3D charts on which none interaction is possible. As [RICARD99] justly notices,

> "the literature on visualization is dominated by the *'sheet of paper'* paradigm".

Following this paradigm, the screen is merely used as a simple sheet of paper, just like 3D visualization does.

On the contrary, Virtual Reality is based on the *'screen'* paradigm. A screen offers much more features than a sheet of paper and consequently, is a perfect medium for Virtual Reality. As a matter of fact, a screen makes possible **interactive** 3D visualization[2].

[2]by the way of panning, zooming, rotating, picking and changing viewpoints.

From now on, the term "3D charts" will refer to charts displayed by 3D visualization whereas "VR charts" will refer to charts visualized in a virtual environment. In this thesis, our field of interest is essentially VR charts.

3.2.4 Right-handed system

In literature, various ways of representing three-axes system are usually used. So as to remain consistent through this thesis, we have decided to make always use of the same system : the "right-handed" coordinate system.

It is made of the following elements :

- The horizontal axis (X-axis) : looking from the front (origin), positive x values are on the right side of the origin and negative ones are on the left.
- The vertical axis (Y-axis) : looking at the origin (0, 0, 0) positive y values are up and negative ones are down.
- The depth axis (Z-axis): positive values are on the nearest side (from origin to the screen) of the axis and the negative ones are on the farthest side.

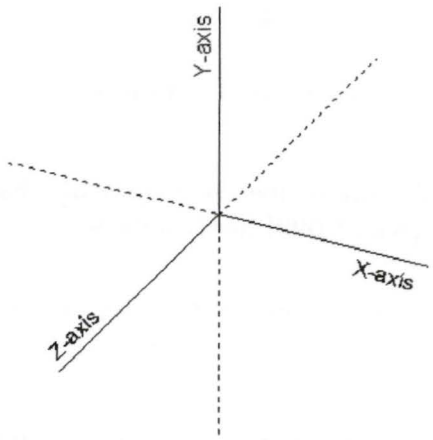


Figure 3.3: The right-handed system

3.2.5 Data dimension vs chart dimension

Finally, an important distinction between *data dimensions* and *chart dimensions* will conclude this section.

Data dimensions are structural units of information, i.e. a list of members. All members are of a similar type in the user data perception. Dimensions offer a very concise, intuitive way of organizing and selecting data for retrieval, exploration and analysis. Table 3.1 shows three dimensions :

- **Year** of which members are 1994, 1995, 1996, 1997 and 1998 (qualitative values)
- **Company** of which members are Microsoft, IBM and Oracle (qualitative values)
- **Comparative revenues** (quantitative values)

Company \ Year	1994	1995	1996	1997	1998
Microsoft	0.5	1.7	2	3.8	9.7
IBM	0.5	0.6	1.8	3.6	5
Oracle	0.5	0.8	2.1	4.2	3

Table 3.1: A set of data dimensions

Chart dimensions are the graphic dimensions capable of displaying data dimensions. The point is that a chart does not need to include the equivalent number of dimensions as the data. Subsequently, the same type of chart can visualize the identical data set in many different ways, each with its advantages and disadvantages.

For example, Column Charts in figures 3.4 and 3.5 represent the spreadsheet of table 3.2 in two different ways. Table 3.3 makes a comparison between these two approaches.

3.3 Semantic properties of data

According to us, the semantic properties of data in the perspective of their visualization are the set of properties which defines the nature of data. The properties of the data we have chosen to analyse are:

1. the *data organization*
2. the *data type*
3. the *temporal dimension of data*
4. the *data set size*

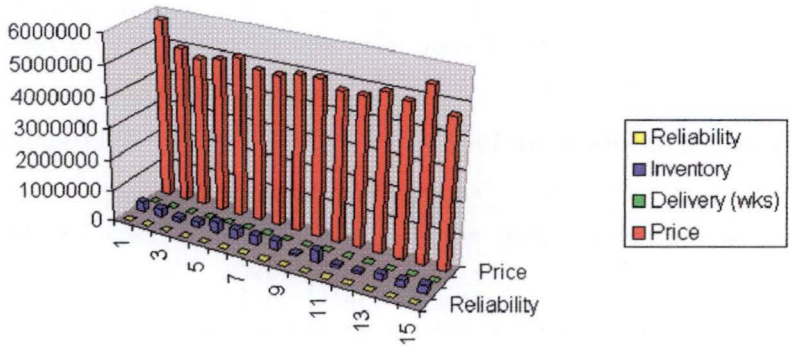


Figure 3.4: First approach of the Column graph

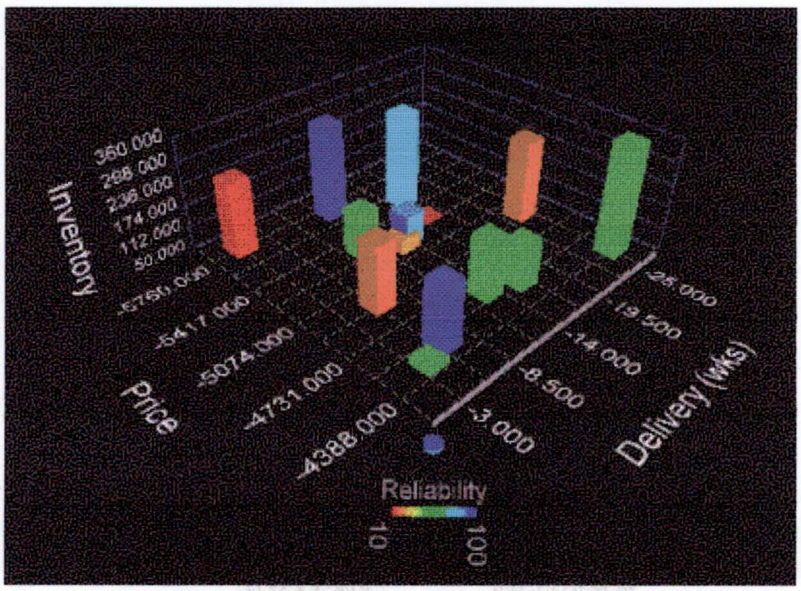


Figure 3.5: Second approach of the Column graph

3.3.1 Data organization

The first semantic property is the *data organization*. We distinguish two main data structures : the *non-geometric* structure and the *geometric* structure.

A *Non-geometric* structure is an organization for which none obvious physical model can be used to represent the data that humans can understand objectively [Zhang-Whinston95] (see figure 3.1, page 35).

Products	Reliability	Inventory	Delivery (weeks)	Price
Product 1:	50	300000	13	5780060
Product 2:	30	250000	15	5000000
Product 3:	80	150000	25	4800000
Product 4:	60	200000	24	4900000
Product 5:	90	360000	8	5100000
Product 6:	100	300000	12	4850000
Product 7:	80	280000	15	4800000
Product 8:	70	320000	18	4950000
Product 9:	30	50000	17	5000000
Product 10:	90	360000	22	4750000
Product 11:	40	100000	26	4750000
Product 12:	80	80000	13	5000000
Product 13:	40	200000	25	4850000
Product 14:	65	200000	25	5500000
Product 15:	49	250000	20	4700000

Table 3.2: the four dimensions of a products series

Geometric organizations are based on physical models. We have picked out three types of geometric organizations :

1. the *hierarchical* organization
2. the *networked* organization
3. the *matrix* organizations

Hierarchical
We define a *hierarchical* organization as a structure based on parent-child relationships. Its purpose aims to establish groups and/or categorize individual elements. Table 3.4 shows a hierarchical organization

Networked
We define a *networked* organization as a structure that allows to explore concepts. Each concept may be connected by links to one or several other ones. This structure can be *cyclic* or *acyclic*.

	First approach	second approach
The four dimensions of the products	Only the z-axis (= one dimension of the chart) is used to represent the four dimensions of the product. Hence four columns are needed to represent the dimensions.	Four distinct dimensions (the three axes and the colour) are used here to represent the four dimensions of the product. Consequently one column can express them all.
What if two products have exactly the same values for the four dimensions ?	No problem since the products are represented along x-axis	**Problem**: the two columns are merging
Scales	**Problem**: only one scale may be used (the y-axis scale). If dimensions have very different values, this hampers the readibility of the graph.	No problem. There are as many scales as dimensions.

Table 3.3: Comparison between the two Column Charts

Population working live in Namur					
Male			Female		
< 25	25-60	>60	<25	25-60	>60
10%	89%	17%	8%	77%	8%

Table 3.4: A hierarchical organization

Semantic networks are a meaningful example. [Schobbens98] defines a semantic network as an oriented acyclic graph. Its nodes represent concepts and its edges represent semantic links between them. For instance, semantic networks fit perfectly to symbolize the human associative memory. Figure 3.6 displays a semantic network.

Matrix

A *matrix* organization allows to represent in a clearer way the data of a relational table in which stand **more** than a **one-for-one correspondance** between the fields [Pilot99]. Matrices are often used in OLAP tools as it is efficient to extract results from complex queries. As a matter of fact, the matrix structure perfectly fits to classify qualitative data. The information in the rows and columns may be in any form including words, numbers, symbols, etc.

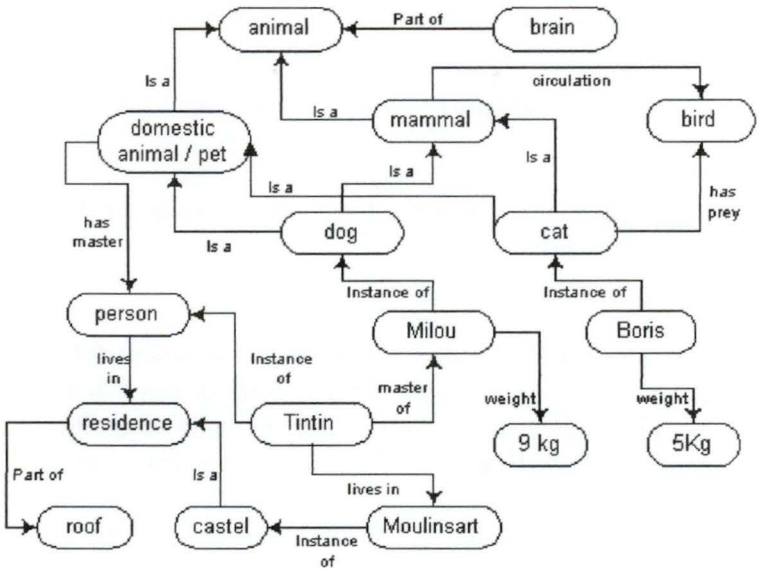


Figure 3.6: A semantic network

	Product		**Service**	
	Physical	**Digital**	**Physical**	**Digital**
Durable	equipment, furniture, clothing,...	a CD, an electronic version of a book,...	repairing,...	digital synchronized clock,...
Short-lived	fresh produce, flowers,...	*N.A.*	hairdresser's	a flight reservation and purchase,...

Table 3.5: A matrix of different product types for e-business

Table 3.5 shows a matrix example.

3.3.2 Data type

The next property is the value *data type*. A data can be either *quantitative* or *qualitative*.

A *quantitative* data is sometimes referred to as a *value*, *intervals* or *numeric data*. Quantitative data are typically made up of entities that have specific numeric values such as heights, weights, ages, etc. Therefore they can be plotted on numbered scales.

A quantitative data can be either *continuous* or *discrete* [Harris99]:

- a *continuous* data is a quantitative data that can take on any value within a given range.
- a *discrete* data takes only specific values and not values in between (for instance, only whole numbers).

A *qualitative* data is sometimes called a *category*, *non quantitative* data or *nominal* data. Qualitative data are typically made up of word descriptions of entities such as people, things, places, etc. The elements of qualitative data can be ordered (e.g. alphabetically, by families,.) but not mathematically manipulated.

Note that if a chart is able to show continuous data by means of an axis with a continuous scale, we consider that the data type is continuous even if the other two axes present discrete scales. For instance, an Area Chart (see section 3.4.1.2) usually presents discrete scale on the z-axis, this axis being used to distinguish the data series. But as x-axis often presents continuous data, we consider that the graph displays continuous data. Similarly, if a chart presents at least one quantitative axis, we consider that the data type is quantitative.

3.3.3 Temporal dimension of data

We use the *temporal dimension of data* in two ways.

The first one is for **making comparisons** between several data at different moments in time (figure 3.7).

The second one is used to highlight **trends** of a data along a time period (figure 3.8). Note that the temporal dimension can be either continuous (e.g. a passing day) or discrete (e.g. five distinctive years).

3.3.4 Data set size

The *data set size* is the **length of a data series** or the **number of data series** in a chart. Table 3.6 sums up those four properties as well as the different categories they can take on.

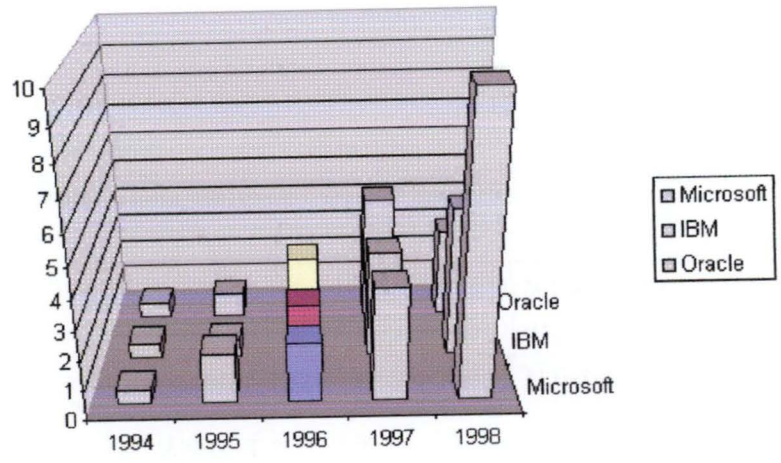


Figure 3.7: Comparison between organizations for a specific year

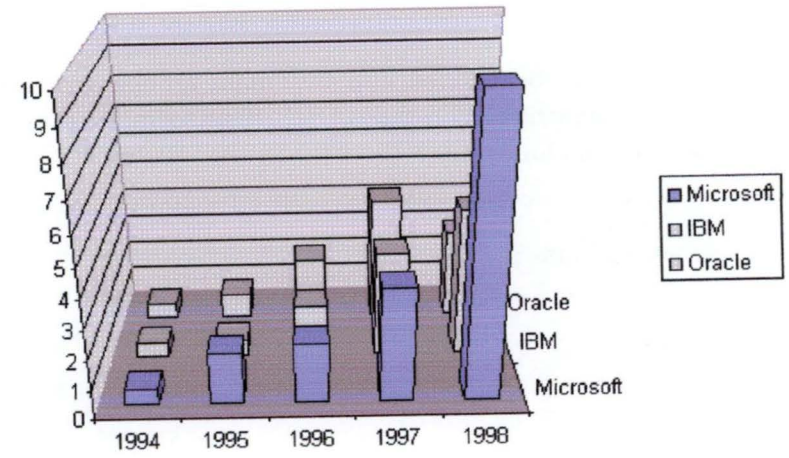


Figure 3.8: Visualization of a trend along a time period

3.4 Presentation of the most common charts

The problem that we have to deal with is the selection of the right chart by the business user. This problem was raised by BARBARA MIREL in the introduction.
As we consider that the selection of the right chart is function of the semantic properties met by each chart, we have to be acquainted with the charts properties. Therefore, we are going to paint a picture of the most common VR Charts[3] by means of empirical observations. The resulting pictures set will

[3]In a clarity concern, we had to limit ourselves to the most common charts among the available VR charts.

Semantic properties	Description
Data organization :	- geometric : hierarchical, networked, matrix - non-geometric
Data type :	- quantitative : discrete, continuous - qualitative
Temporal dimension of data:	- comparison - trends
Data set size:	- amount of data to visualize

Table 3.6: Summary of the semantic properties

allow to infer a summary table of the semantic properties met by each single chart (see figure 3.6).

The charts presentation will be divided into two main groups according to the data organisation (geometric or non-geometric). This semantic property is according to us the most fundamental.

The non-geometric set includes :

1. the Column Chart
2. the Area Chart
3. the Scatter Chart
4. the Ribbon Chart
5. the Surface Chart
6. the Pie Chart
7. the Map Chart.
8. the Temporal Star
9. the Combination Chart
10. the Animated Chart
11. the World Chart

The geometric set includes :

1. the InfoBall
2. the Giotto 3D
3. the Information Cube
4. the DataCube

3.4.1 Non-geometric Charts

3.4.1.1 Column Chart

Visual description:
A *Column Chart* displays discrete quantitative information by means of a series of geometric shapes such as rectangles, cylinders, etc. Each column represents a data element, and a complete set of columns represents a data series. Columns may be lined up side to side as well as front to back.

Function:
Column Chart is particularly efficient in the following roles:

- comparing multiple items at various points distributed along a time period
- showing how relationships between multiple items change with time
- looking for relationships between multiple data series
- condensing onto one graph either several single series already displayed in three-dimensional charts or data series of which visualization could require several two-dimensional charts.

Limitations:
Even if there is no technical limit concerning the *number of data series* or the *series length* displayed by the graph, we have observed practical limits above which the graph becomes confusing.

Considering the *number of data series* generally distributed on z-axis, we have observed in most cases that the magical number[4] (= 7 +- 2) is an appropriate limit. The magical number expresses the cognitive limit of the human mind. It means that more than ten data series will worsen charts readibility.

[4]G. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information", *Psychological Review* 63, no. 2 (1956): 81-97

The *length of the data series* depends on the number of data series displayed on the chart and is limited by the necessity to keep a global clear and not confused view of the chart.This way, the less data series are displayed, the longer these series may be, and vice versa. Note that these limits are suitable for the other charts described in this section.
We can justify this choice by the "limitations of the graphical interface" explanation in [Ricard99] :

> "Our visual sense has a rather small area of high focus. A problem arise when a user must concentrate on the visual feedback from one part of the display, so that feedback from another part of the display may be missed as it is outside the area of visual focus. As the amount of information contained by the visual display becomes bigger, the user can become overloaded and the display ineffective".

Since the Column Chart is a VR chart, this limitation is pushed back thanks to the possibilities of interactivity[5]. Interaction may allow to generate clear and not confused subsets of the graph which focus on particular details.

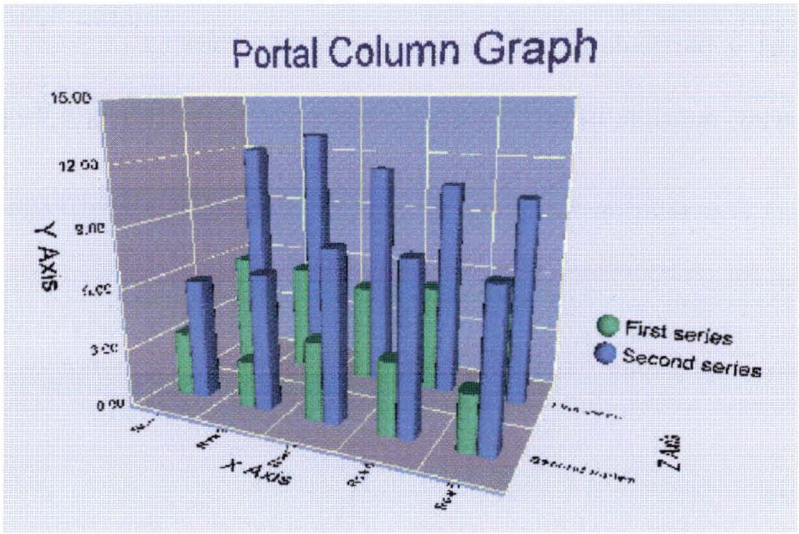


Figure 3.9: A Column Chart

Variant forms:
Eventually, Stacked Column Chart may be used to indicate proportion of each column segment in a whole. In this case, each column is a composition of columns from several series of data. Each column segment is coloured to indicate the series to which it belongs. In figure 3.10, Columns have a fixed size and therefore do not represent an absolute sum of its composants.

[5]e.g. drilldown, filtering, sorting, queries, zooming, etc.

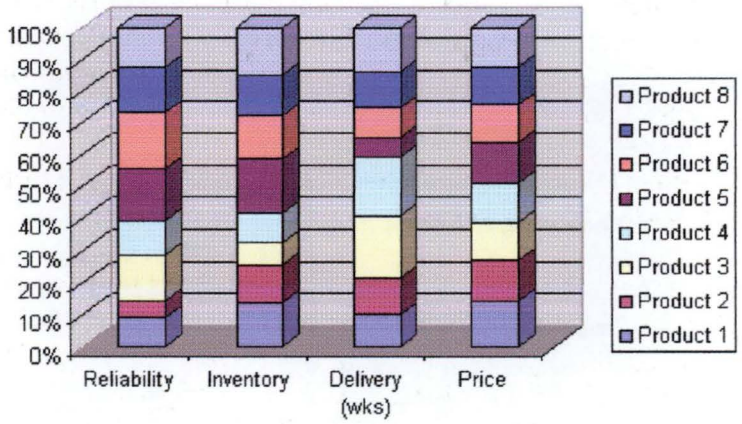


Figure 3.10: A Stacked Column Chart

3.4.1.2 Area Chart

Visual description:
An *Area Chart* is composed of block-like shapes with jagged edges along the tops representing the boundaries of continuous quantitative data. The data values are particularly visible thanks to the full area under the edges. Adding the transparency to this chart would allow the user to look better through areas. Furthermore, interactivity allows the user to select and pull a piece of the chart out in order to analyse it more comfortably.

Function:
The Area Chart is suitable at underlining the trends of information along a time period as well as conveying comparisons of areas.

Limitations: /

Variant forms: /

3.4.1.3 Scatter Chart

Visual description:
Also called "Point Chart", the *Scatter Chart* displays discrete quantitative information by means of points (represented by spheres or cubes placed in the chart). Three-dimensional scatter graphs generally have quantitative scales on all three axes.

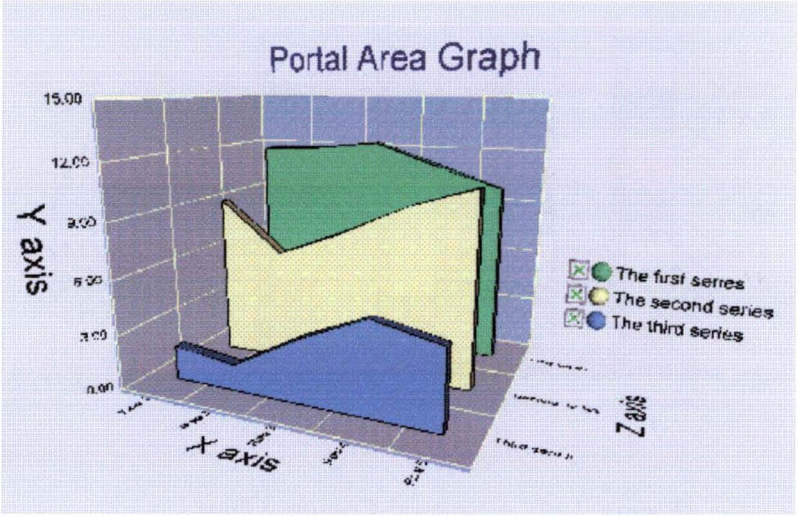


Figure 3.11: An Area Chart

The Scatter Chart possesses six dimensions to display information. These dimensions are:

- the three coordinates (x, y and z),
- the colour,
- the transparency and
- the shape size[6].

A seventh dimension may be the time which can be represented through animation (see section 3.4.1.10).

Function:
A Scatter Chart is used to correlate multiple business inputs in order to identify trends, make comparisons and find connections hidden within data.

Limitations:
As its function is only to spot the presence of outlying data within a set, it seems that this chart offers a more flexible limit on the number of data series or set on the graph than the other charts.

Variant forms: /

[6]the radius for a sphere or the edges length for a cube

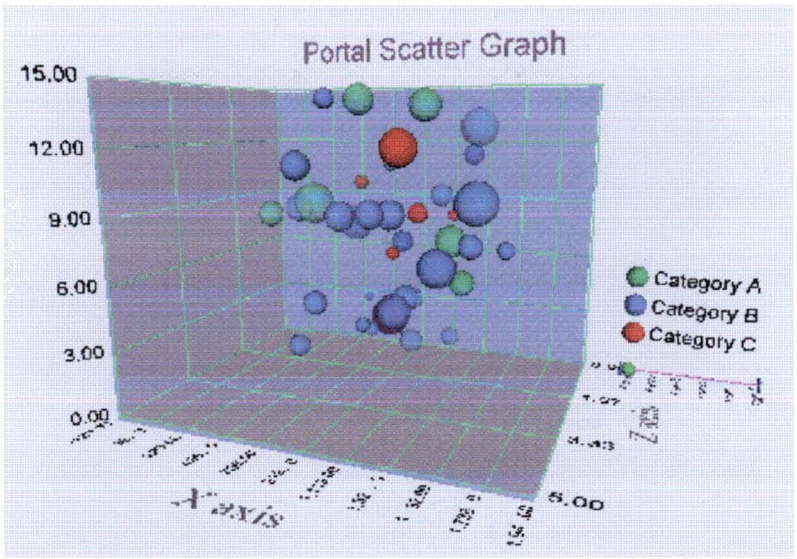


Figure 3.12: A Scatter Chart

3.4.1.4 Ribbon Chart

Visual description:
The *Ribbon Chart* displays continuous quantitative information by means of thick lines, visually similar to ribbons. Typically, it has a quantitative scale attached to the vertical axis and a qualitative or quantitative scale on the other two axes depending on the data nature.

Function:
The Ribbon Chart is particularly useful to underline the data trend along a time period and highlight the evolution of data.

Limitations:
This graph features the same practical limit of the data series number than the Column Chart. The problem of the perspective view in three-dimensional space makes the ribbons of the Ribbon Chart particularly difficult to compare. This difficulty is partially solved by using an Area Chart (see section 3.4.1.2)which the opaque surfaces help to comparisons.

Variant forms:
Ribbons may be used to join some points of the Scatter Chart in order to highlight the relationships between them. This new chart is called Scatter Ribbon Chart. It benefits from the advantages of the Ribbon Chart as well as

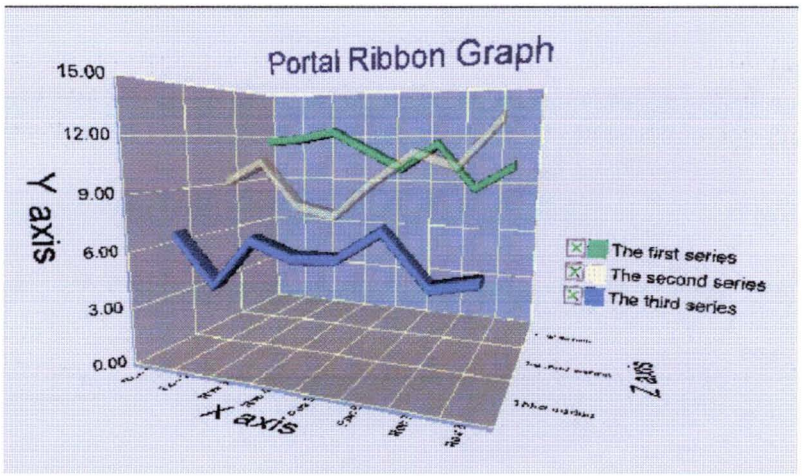


Figure 3.13: A Ribbon Chart

those of the Scatter Chart (see section 3.4.1.3): it shows a global evolution[7] and it draws user attention to some more important data[8].

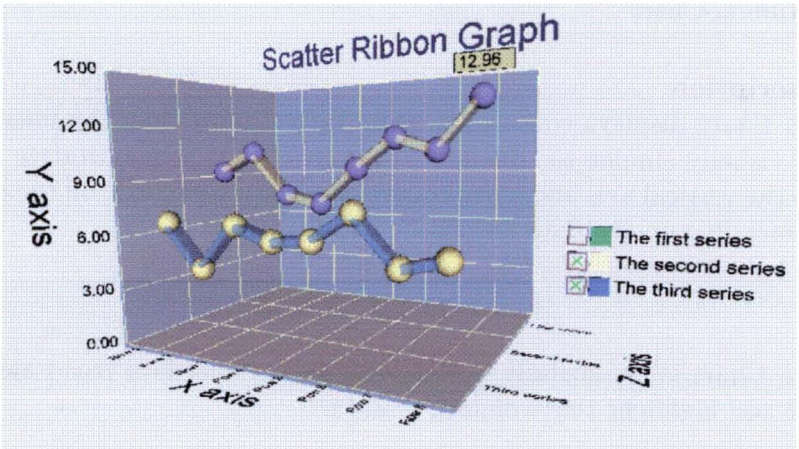


Figure 3.14: A Scatter Ribbon Chart

3.4.1.5 Surface Chart

Visual description:
A *Surface Chart* is a Wireframe Chart (a chart consisting of a series of lines that simulate the surface of a data graphic [Harris99]) in which the areas between the lines are opaque. This chart is really appropriate to display continuous quantitative data.

[7] with the edges of the Ribbon Chart
[8] with specific points of the Scatter Chart

Function:
The wireframe structure makes the extreme points and patterns of data stand out clearly. A Surface Chart is very useful to highlight trend of large data sets or making data comparison when combined with a Combination Chart (see section 3.4.1.9).

Limitations:
Although vertical scales on these charts are generally quantitative, in most cases it is difficult to determine actual values accurately. For this reason, these charts are often used to indicate the general nature of the data rather than supplying detailed information.

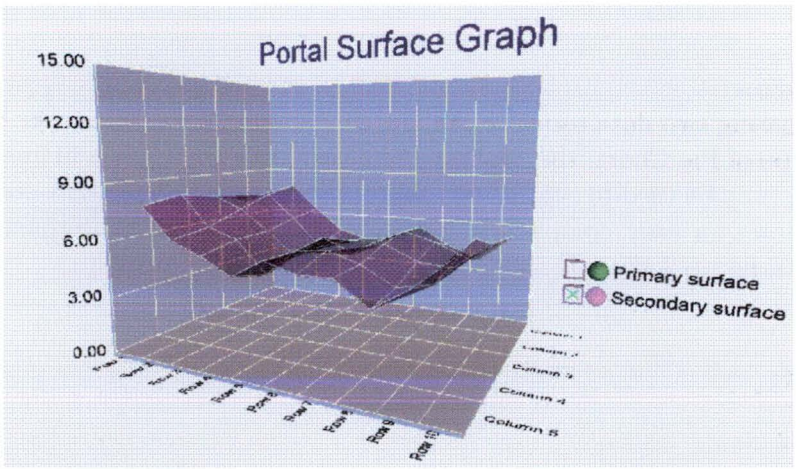


Figure 3.15: A Surface Chart

Variant forms: As mentioned just above, this chart may be associated with another (e.g. a Column Chart, a Ribbon Chart, a second Surface Chart, and so forth) to form a Combination Chart (see section 3.4.1.9).

3.4.1.6 Pie Chart

Visual description:
The *Pie Chart* is a member of the Proportional Area Chart family. Its purpose is to show the proportion of parts to a whole (therefore, it represents discrete quantitative data). A Pie Chart consists of a circle divided into wedge-shaped elements. The height of each wedge may vary either to convey additional dimension or just for a cosmetic aim. The area of each slice reports percentages of a total amount represented by the Pie Chart. This percentage is represented by three ways : the angle of the segment, the area of the segment and the length of the arc. See figure 3.16

Function:
This chart is the best means to synthesize a large amount of data in order to compare data categories. The segments are indiscriminately arranged according to the desired criterion of the user (for example, from the less to the most important percentage), with the clockwise direction normally used. To emphasize segments of the chart, the user can use colours, transparency, glow or other visual cues, but also drag a slice out of the body of the Pie Chart. Note that wedge height may also be used to add more data dimensions to a single Pie Chart.

Limitations:
To obtain a readable Pie Chart, the data have to be gathered together in a reasonable amount of slices.

Variant forms:
When comparing two data series or changes in the same data series over time, a second type of Pie Chart, the *Stacked Pie Chart* (two Pie Charts fitting into each other), may be used to replace two charts shown side-by-side. The stacked strucure allows an easy comparison between wedge-shaped counterparts, as illustrated in figure 3.17.

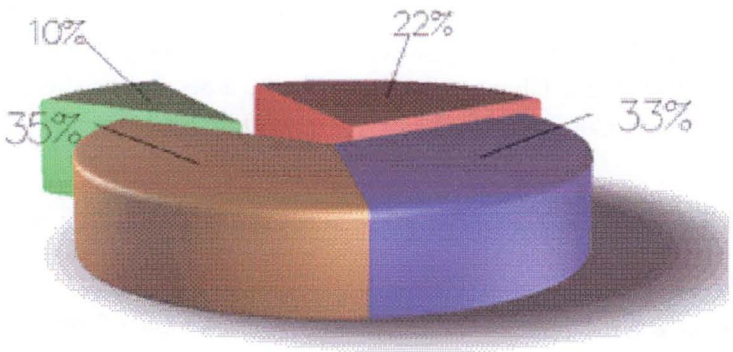


Figure 3.16: A Pie Chart

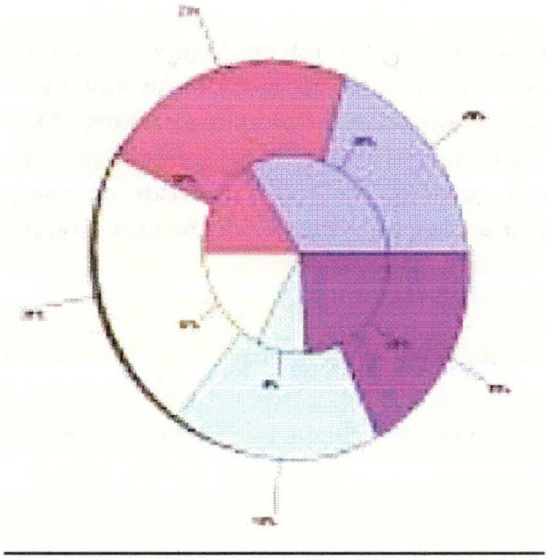


Figure 3.17: A Stacked Pie Chart

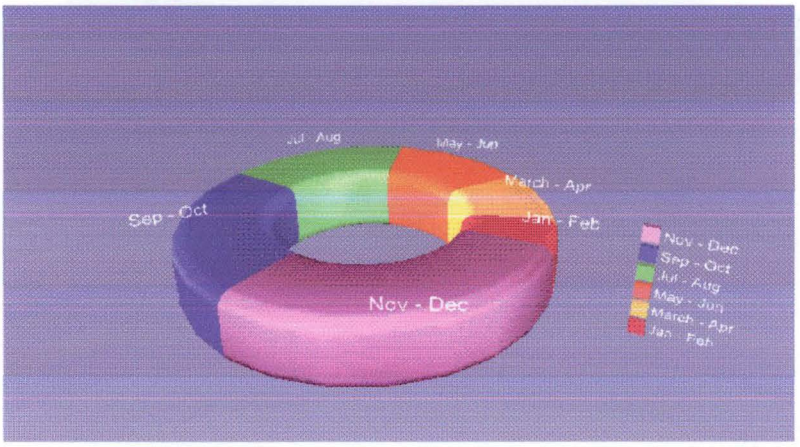


Figure 3.18: A Donut Chart

Figure 3.18 shows another current Pie Chart: the Donut Chart.

3.4.1.7 Map Chart

Visual description:
A *Map Chart* represents a classic geographic map in which each area is extruded in such a way that the extrusion shows an information concerning the geographic region. A Map Chart allows the user to extrude each cell of the chart to improve the data visualization and reveal additional discrete quantitative information not present in a two-dimensional visualization.

Function:
A Map Chart is a means to display information in function of its physical location. Each element of the map is raised in proportion to the value it represents, allowing comparisons with neighbouring states. This representation can be more appropriate than other charts in some cases. For instance, it is easier for the user to locate and recognize the state on the map instead of labelling each column of a Column Chart with the state name.

Limitations:
Although the vertical scale of a map is quantitative, it is difficult to determine actual values from the map accurately. For this reason, a Map Chart is generally used to indicate the relative difference between the various elements of a map.

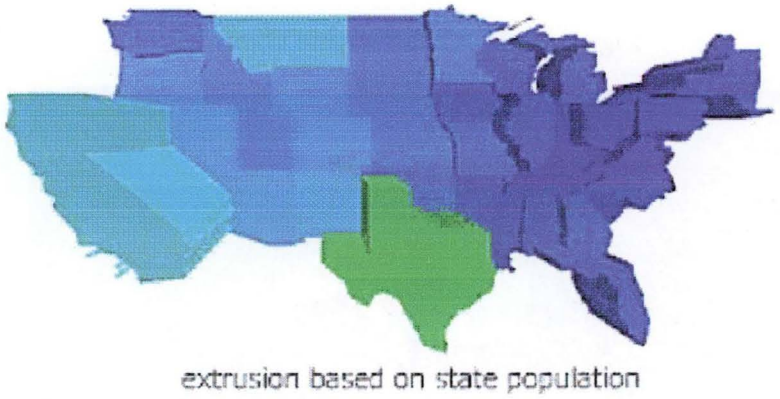


Figure 3.19: A Map Chart

Variant forms : /

3.4.1.8 Temporal Star

Visual description:
A *Temporal Star* is a set of *Simple Stars* linked by a central vertical axis representing time [Noirhomme00a].

A *Simple Star* is a circular graph with three or more radial axes distributed equally around the 360 degrees of the circle and representing statistical variables. All axes have either the same scale or a specific scale. Data elements are plotted on the axes. A quantitative variable (average or interval) is represented by a graduated axis and a qualitative one is represented by dots equally distributed on an axis.

Function:
A Simple Star is used for comparative purpose by reading the actual values of data elements plotted on the axis. The aim of the Temporal Star is to visualize a set of data elements at different moments of time. Each moment is represented by a Simple star. In order to emphasize the evolution of quantitative variables from one time moment to another, the interval extremities (min, max) or means may be joined by a semi-transparent veil.

This chart possesses an interesting feature not discussed up till now. When selecting an axis, a **sound** will be played indicating the type of the axis (e.g. quantitative or qualitative). Since this thesis focuses only on visual features of Virtual Reality, the use of sound will not be developed. Nevertheless, the sound cue is discussed in details in [Ricard99], [Noirhomme00a] and [Noirhomme00b]
.

Limitations:
According to [Ricard99], a Simple Star[9] allows about 16 variables (i.e. 16 axes) to be displayed. And even if this number can be raised out, progressively, it leads to an overload of the graphic.

Variant forms: /

The three following charts are more complex because they are compositions of the previous ones. They display multiple data series, using two or more types of charts to represent them.

3.4.1.9 Combination Chart

Visual description:
A *Combination Chart* is the combination of a Surface Chart with another chart.

The most often used combination is the *Surface/Column Combination* Chart (see figure 3.21), which is a fog-like surface superimposed over columns, giving the appearance of a cityscape emerging through the fog. Depending on the charts combined, the data are discrete or continuous.

Function:
This chart is excellent to compare two data sets , e.g. expected results or threshold (fog) in relation to actual results (columns, areas, and so forth).

[9]called a *3D Zoom Star* by [Ricard99].

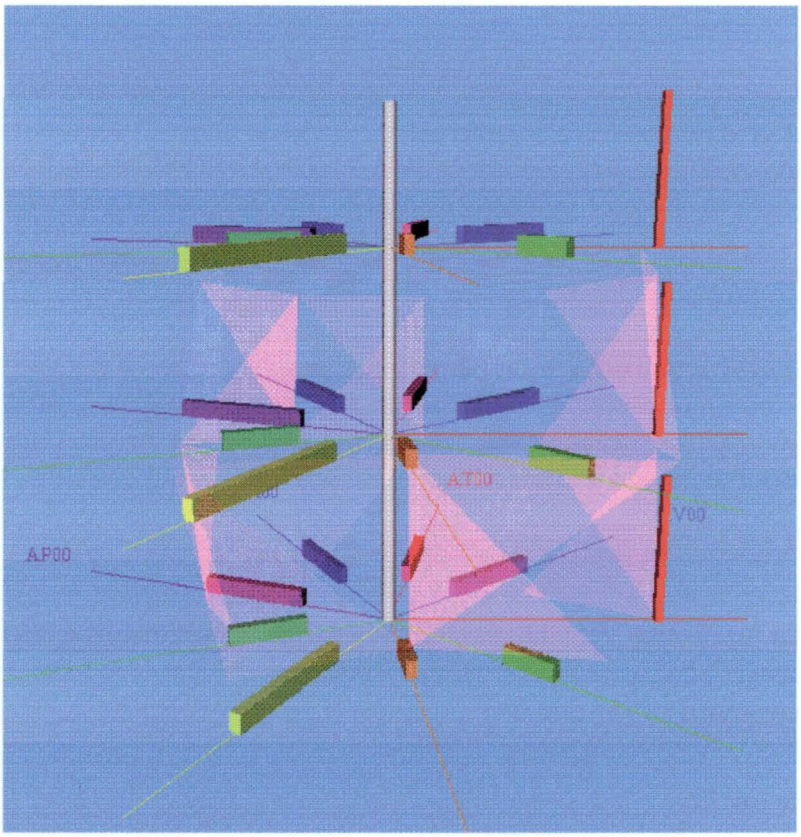


Figure 3.20: A Temporal Star

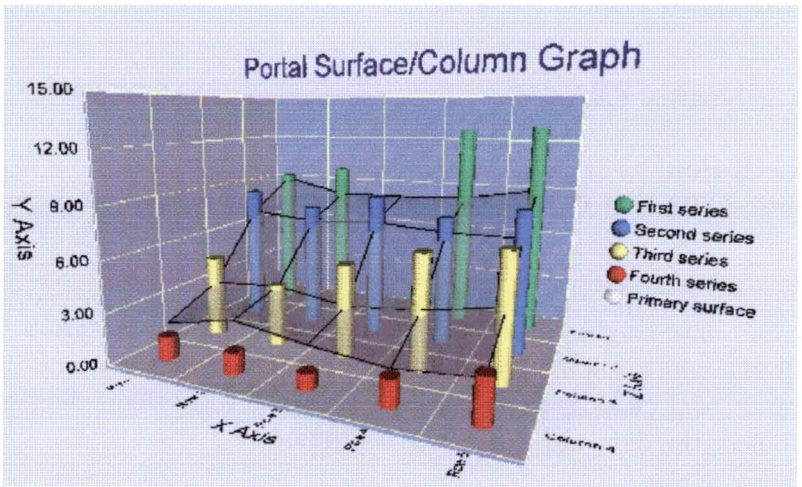


Figure 3.21: A Surface / Column Chart

Limitations:
The limitations of this chart depend on the limitations of its component charts.

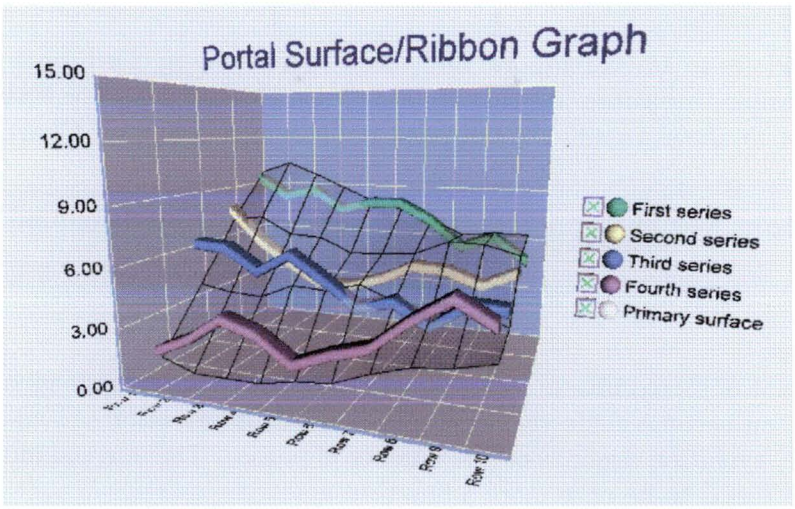


Figure 3.22: A Surface / Ribbon Chart

3.4.1.10 Animated Chart

Visual description:
An Animated Chart is a particular chart which collects only a set of representations of the same chart at different moments of its evolution. Technically, time is represented by storing various phases of a chart and then by animating them.

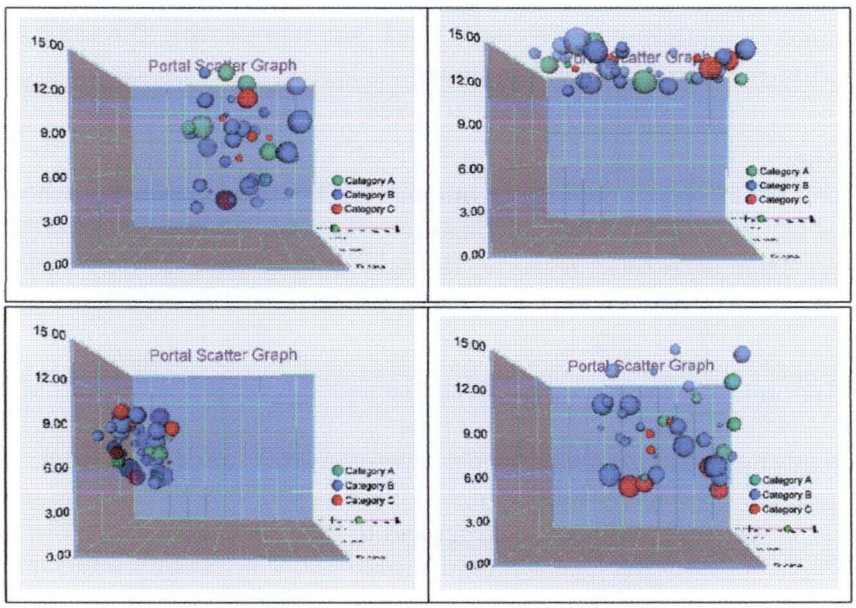


Table 3.7: Four phases of a Scatter Chart

Function:
This chart allows to add a temporal dimension to an existing chart when it is no more possible to incorporate this dimension by another way. For instance this mechanism allows to add the temporal dimension to a Scatter Chart.

Table 3.7 shows four phases of a Scatter Chart displaying the evolution of a data set.

Limitations:
As the amount of stored phases increases, the user can become overloaded and the display can become ineffective. The clarity of the information provided by the temporal evolution depends on the number of phases. The magical number $(= 7+\text{-}\ 2)$[10] seems to be once more a fine threshold.

3.4.1.11 World Chart

Visual description:
A *World Chart* is a virtual environment composed of various charts (either geometric or non-geometric) collected and classified according to a qualitative criterion (e.g. a name, an order number, etc).

Function:
A World Chart is an environment in which user can travel through to reach and examine the charts contained in this world. It is advised to collect charts related in some way, e.g. by the same dimension. For instance, a world can be an environment containing twelve charts, each one representing the same kind of information **for a month of the year**.

Limitations:
The problem could be the navigation among the different charts of the world, but thanks to predefined navigation features, the user can reach a chart in a simple way. Thanks to these features, the amount of charts in the world and consequently, the amount of data set, may be large.

3.4.2 Geometric Charts

A *geometric* chart is a physical model that can be used to represent data and that humans can understand objectively. Our geometric charts represent hierarchical, networked and matrix structures in 3D. They are designed to help the user to navigate between concepts, compare nodes or classify information. Consequently, these charts are efficient to convey qualitative data.

[10] see page 43

Figure 3.23: A World Chart

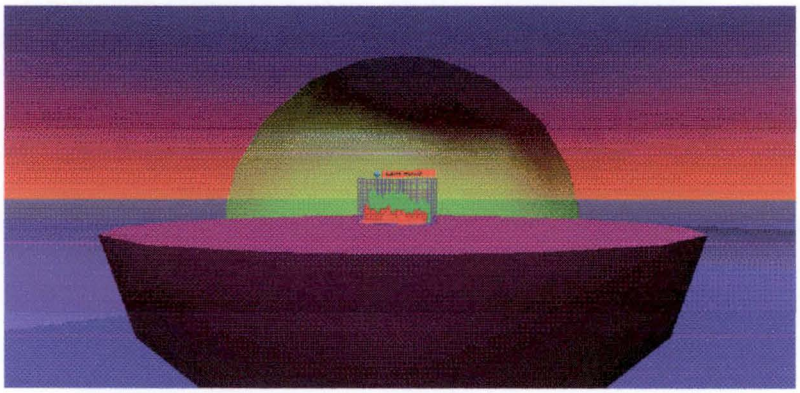


Figure 3.24: A chart within a World Chart

The presence of navigational operations such as rotation, translation, and zooming not only help the user to understand the structure, but also allows a more effective use of the screen space since only a small amount of information is displayed at any time.

We have picked out four different geometric charts :

1. the InfoBall
2. the Giotto3D
3. the Information Cube
4. the DataCube

3.4.2.1 InfoBall

Visual description:
The *InfoBall* [Roskothen99] is a set of information spheres connected to each other. Each sphere (or node) of the InfoBall displays a collection of links which can open connections to new spheres. Figure 3.25 and 3.26 display InfoBalls. The user can drag a sphere and rotate it into a new position. Clicking on a link will open a new sphere in the chart while clicking on the "+" sign will close a sphere. Once a node is selected, the previous levels and the sibling nodes are not taken into account anymore. That is why they are much less visible than the selected ones.

Function:
The InfoBall is particularly appropriate to explore concepts and therefore to symbolize *networked structures*. Actually, the structure offered by the InfoBall allows one concept to be reached from many other ones. As the opposite of trees, a sphere link can lead to any other nodes of the chart. Moreover, the InfoBall does not show any levels structure since only one sphere is focused at a time. Consequently all the spheres belong to the one and only level.
Finally a node linked by different sibling nodes still exists even when some sibling nodes are erased from the whole structure. If necessary the Infoball can simulate a hierarchical structure.

Limitations:
As a matter of fact, this chart does not allow to visualize the whole structure at the same time since the sibling nodes are much less visible than the currently selected sphere.

Variant forms:
Sphere links could give access to various media types such as charts, texts, hyperlinks, and so forth. The InfoBall would then be more interesting than simply linking spheres.

3.4.2.2 Giotto 3D

Visual description:
The *Giotto 3D* technique [Garg-Tamassia96] models a hierarchical structure by a three-dimensional Directed Acyclic Graph (DAG). A Directed Acyclic Graph is a graph with directed edges which constructs no directed cycles. Consequently a node may have several parents to which none directed path leads from the node (see figure 3.27)

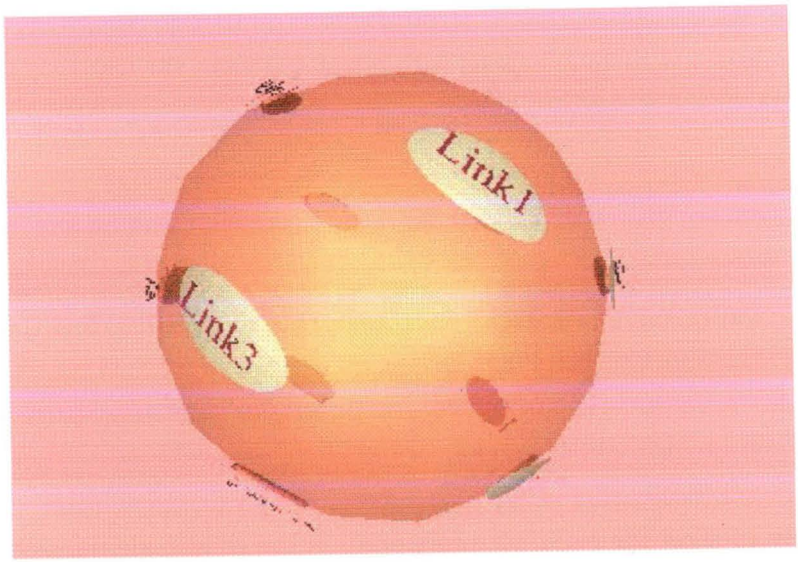


Figure 3.25: The InfoBall

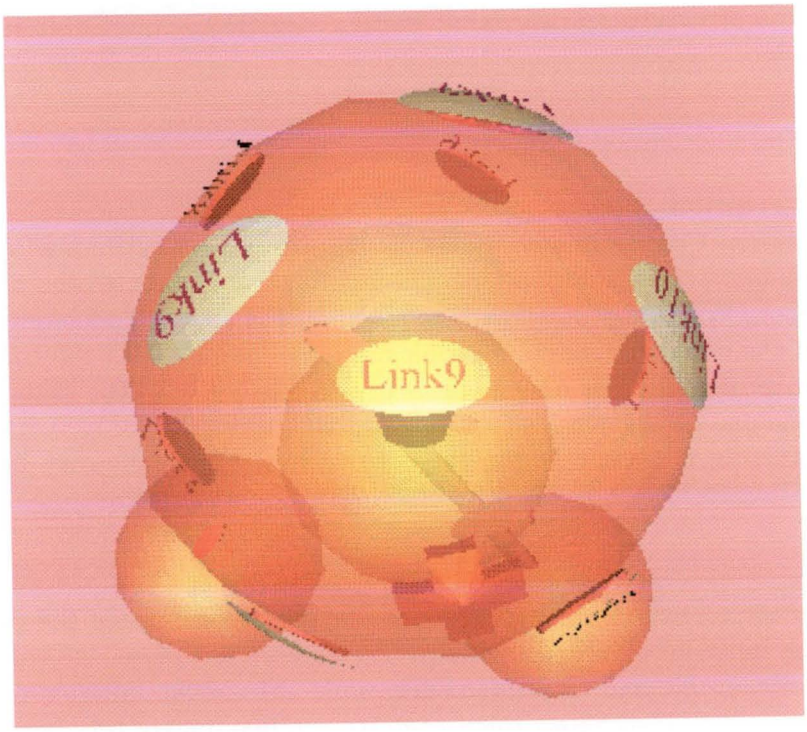


Figure 3.26: An exploded InfoBall

Function:
The treelike structure is particularly well appropriate to visualize a hierarchical structure.

Limitations:
Since the whole structure may be seen within a global view, the user may suffer from the "lost within the forest" effect.

Variant forms:
Besides what Giotto 3D offers, various interesting features could be also developed.

- Firstly, additional information (chart, text, hyperlink, and so forth) could be embedded inside each node[11] .
- Secondly, in order to lessen the "lost within the forest" impact, targeted nodes can be highlighted by glow or light.
- Finally, even the edges could represent data. For instance, using transparency to show the certainty probability of the child node information.

As a matter of fact, we have implemented this new features in our own Giotto3D, that we called **Tree3D** (see figure 3.28). As its name suggests it, the Tree3D features a tree structure, i.e. a DAG of which nodes have one and only parent [Schobbens98].

3.4.2.3 The Information Cube

Visual description:
The *Information Cube* technique [Rekimoto93] uses translucent, nested cubes to represent hierarchical information. The Information Cube uses the nested box metaphor. Practically no effort is required for the user to understand the meaning of the visualization, since people are quite familiar with the concept and the usage of a box - as a container - in their daily lives. The outermost cube corresponds to the top level data, and going down to the next level is made by entering in a cube within the current one. Each cube may bear a label indicating its level, the displayed information, and so forth.

Function:
The semi-transparency of each cube allows the user to see inside the cubes, while hiding inner information gradually. As a result, the closer information is more easily visible and understandable while the deeper information remains quite visible. That allows the user to see and manipulate very huge and complicated hierarchical structure, because the screen image complexity is always maintained reasonable. Transparency permits not only to see next levels through a cube but also shows the additional information possibly embedded in a terminal cube (such as chart, text, and so on). A terminal cube will be opaque if it does not countain embedded information.

[11] represented by a sphere

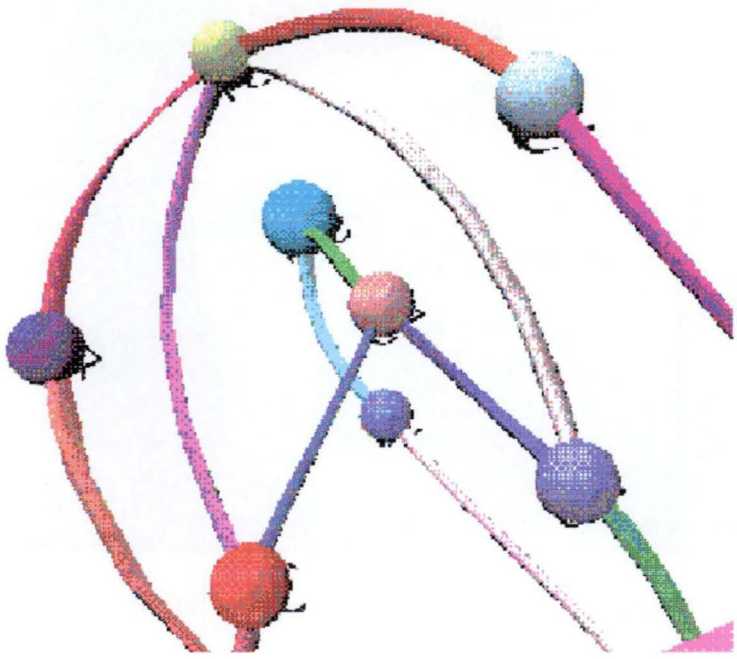


Figure 3.27: A Giotto3D

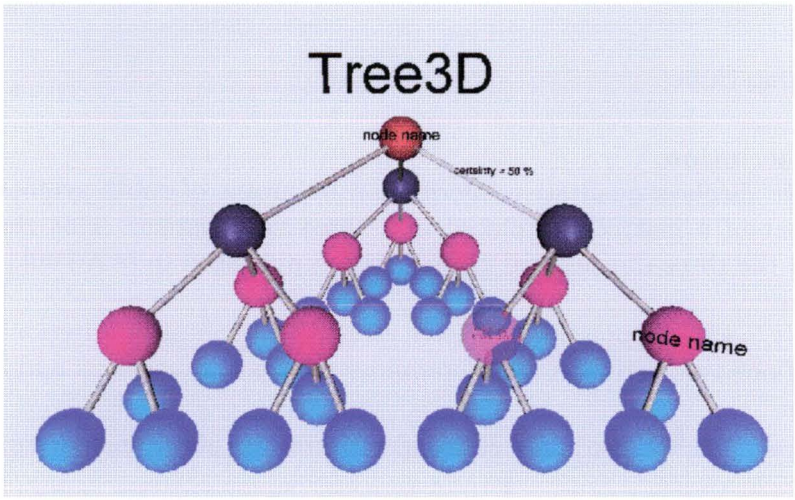


Figure 3.28: A Tree3D

Limitations:
The shortcoming of the Information Cube is the difficulty for the user to locate the current focused node over the whole hierarchy.

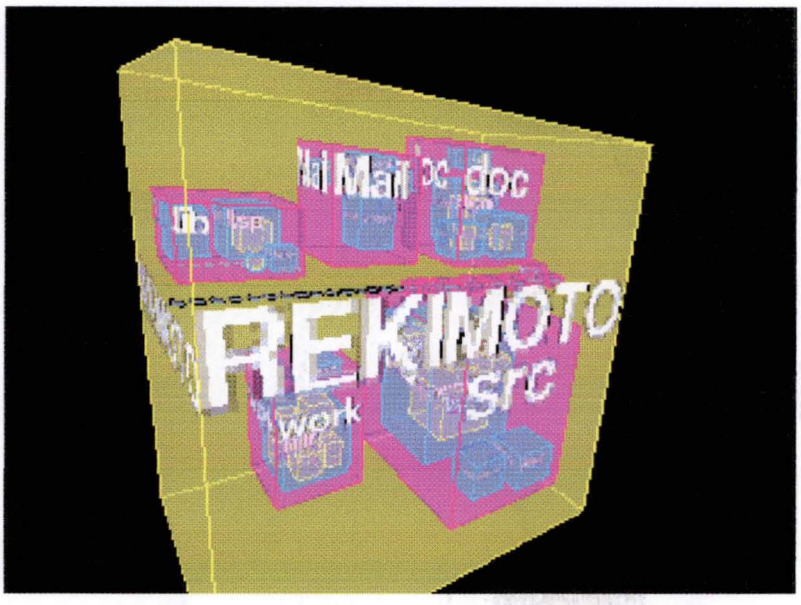


Figure 3.29: Displaying a Unix directory using the Information Cube

Variant forms:
To overcome the limitation, a *glow* effect could be implemented, allowing the highlight of a node researched by the user.

3.4.2.4 DataCube

Visual description:
A *DataCube* displays qualitative information by means of cubes symbolizing each element of a matrix and placed in a three-axes system. This chart is inspired by the three-dimensional representation published in [AWT00].

Function:
Functionally, each axis presents exclusive values of a qualitative criterion. The selection of one value on each axis determines a cube that contains additional information.
A popup area (the ? icon) on a cube indicates the set of criteria determining this cube while clicking on the i icon leads to the information contained by the cube.

Limitations: /

Variant forms: /

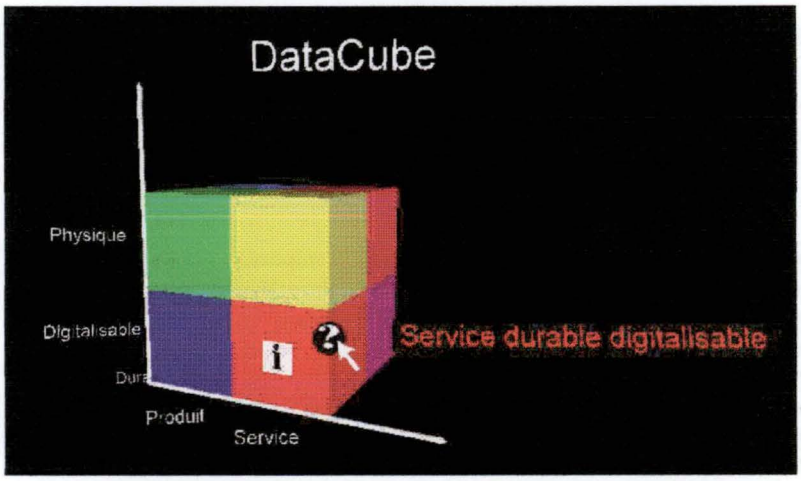


Figure 3.30: A DataCube

3.5 Conclusion

In this chapter, our purposes were to present some semantic properties of business data in the perspective of their visualization, and to present the most common VR Charts.

These purposes help us to solve the problem raised by BARBARA MIREL, i.e. the selection of the right chart by the business user.
As we consider that the selection of the right chart is function of the semantic properties met by each chart, we have infered a summary table of these semantic properties (see figure 3.6).

In chapter 4, we shall examine how far the Virtual Reality technology can be an efficient platform to deal with the challenges raised by Business Information Visualization. This is the the core chapter of this part.

Chart name	Data organisation		Data type	Temporal dimension of data	Size of the data set
Column Chart	non-geometric		discrete quantitative	comparison and trends	limit in number of data series
Scatter Chart			discrete quantitative	comparison and trends	medium amount of data
Ribbon Chart			continuous quantitative	comparison and trends	limit in number of data series
Area Chart			continuous quantitative	comparison and trends	limit in number of data series
Surface Chart			continuous quantitative	comparison and trends	large amount of data
Pie Chart			discrete quantitative	comparison and trends (Stacked Pie)	large amount of data
Map Chart			discrete quantitative	comparison	limited by the map
Temporal Star			quantitative or qualitative	comparison or trends	maximum of about 16 variables by Simple Star
World Chart			qualitative		large amount of data
Combination Chart			discrete or continuous quantitative	comparison and trends	depending of the charts combined
Animated Chart			depending of the chart animated	comparison and trends	large amount of data
InfoBall	geometric	networked / hierarchical	qualitative		large amount of data
Giotto 3D / Tree3D		hierarchical	qualitative	comparison	large amount of data
Information Cube		hierarchical	qualitative		large amount of data
DataCube		matrix	qualitative		visual limit

Figure 3.31: Summary of the semantic properties

Chapter 4

Assets and Shortcomings of Virtual Reality in BIV

This chapter is the core of the first part.

We shall examine how far the Virtual Reality technology can be an efficient platform to deal with the challenges raised by Business Information Visualization. For this purpose, we shall discuss the key assets and shortcomings of the Virtual Reality technology for the abstract representation of business data.

4.1 Key assets

4.1.1 Introduction

In studies conducted by Barbara Mirel [Mirel99] with retail market analysts, those analysts estimated that 60% to 80% of their analytical time is taken up formatting data and moving back and forth between spreadsheets and graphics. That leaves only 20% for the essential part of data analysis - interpreting, concluding, and developing new strategies. These studies prove that business analysts need better ways to analyse data visually.

Interactive visualizations offer this improvement. Interactive visualizations are

- **dynamic**, letting the user interacts with multiple, linked views at once to see the same data from different perspectives.
- **interactive**, allowing the user brings in and changes additional dimensions through visual cues such as colour, size and position.

- let the user makes queries by **directly manipulating** data in graphics - selecting, deleting, sorting, filtering, zooming.

Briefly, the user almost simultaneously searches for data, retrieves them and interprets them.

We are now looking further into the assets of the Virtual Reality technology as a Business Information Visualization tool.

4.1.2 Visual User Interface and interactivity

Information-rich user interfaces are becoming increasingly important as computer technologies are used more to give information than to automate. The critical aspect of designing such interfaces is to support effective interaction between human and information.

According to Ping Zhang [Zhang99b], any decision-making support system or problem-solving support systems should be developed from a human-centered perspective. A human-centered visualization system should therefore help the user to achieve cognitive effectiveness and efficiency by shortening cognitive distance from visual representations and removing mediation for thinking .

The "Visual User Interface" (VUI) is a method of direct manipulation enabling the user to take a more active role in the process of visualizing and investigating data. The most important difference between a traditional Graphical User Interfaces (GUI) and VUI is that the user interacts directly with the on-screen graphics and the data behind the graphics without having to work with traditional GUI controls such as pull-down menus, dialogs, or buttons. Consequently, the main asset is that the on-screen graph, chart or data representation **is central to the user's focus**.

Briefly, the representation "plays the part" of the interface, resulting in the commitment of fewer cognitive resources.

The Virtual Reality technology makes VUI possible. VR allows the user to actually interact with the data and let him explore them. Many interactions such as panning, zooming, rotating, picking, changing viewpoints and drilling-down

- let the user quickly perform more thorough data analyses

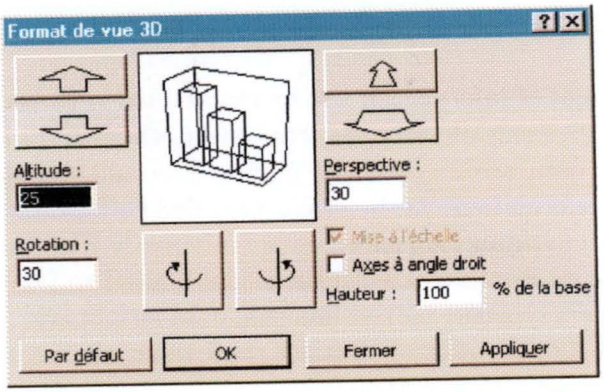


Figure 4.1: A Graphical User Interface

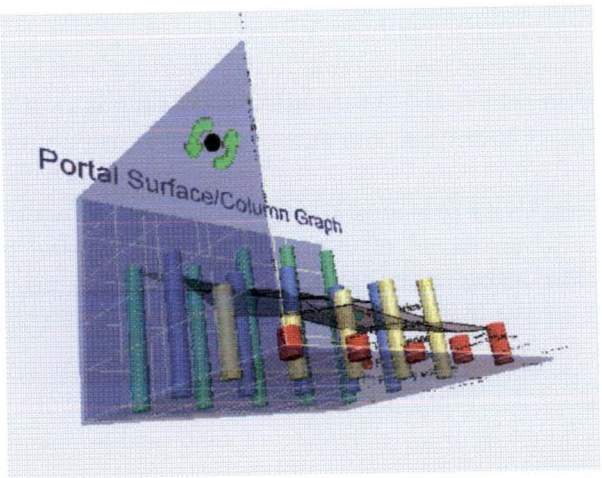


Figure 4.2: A Visual User Interface

- gain new perspectives
- facilitate iterative queries
- drill down from summary to detail data.

These interactions are favourable to the observation of trends, relationships, distributions, outliers, anomalies, and so forth.
For instance, by clicking on a visible part of the graph, it can respond by presenting additional information, perhaps forming a new database query, for better understanding. Clicking on a chart can explode a segment to highlight some information. Of course, although VUI offers an appealing paradigm, we consider its efficiency will be undoubtedly increased when exploited in conjunction with GUI.

4.1.3 Best cost-effectiveness of the screen space

A very limited amount of information can be represented with two dimensions whereas the depth obviously allows a best cost-effectiveness of the screen space: large quantities of data are in this way manageable and multi-dimensional representations of information can be easily provided to business users. This is valuable because multidimensional views are inherently representative of an actual business model. Rarely is a business model limited to fewer than three dimensions. Managers typically look at financial data by scenario (for example, actual vs. budget), organization, line items, and time; and at sales data by product, geography, channel, and time.

Therefore, a crucial point has to be raised. The depth allows a best cost-effectiveness of the screen space only in VR charts.
Large data sets are totally unmanageable in 3D charts where backwards data are inaccessible and in which the informational content is overloaded by the excess of information. Charts become subsequently unusable and difficult to interpret.
At the opposite, interactivity provided by Virtual Reality allows manipulations of the graphs but also creations of various meaningful views from a single voluminous chart. For instance, dynamic queries allow the user to focus his attention on particular details[1].

Nevertheless, this great flexibility of Virtual Reality will be only possible with particularly well designed interfaces. Design of such usable interfaces is unfortunately a real challenge.

4.1.4 Visual cues of Virtual Reality

Since our senses have developed for thousands of years in a physical world, we explore and understand reality in physical terms. Geometric shape, dimension, position, colour, motion and other attributes are the signs of reality, which convey its information most efficiently. Our ability to perceive information in numerical or textual form is not as strong, and that is the reason why people find ways to analyse data visually [Dimension00].

The following features of VR can enhance data visualization :

- Glow, colour, shape, size, texture and transparency may be used as dimensions of their own for conveying data relationships. These features will be discussed in chapter 5.
- Data may now be animated to convey time (see section 3.4.1.10)

[1] thanks to the selecting, deleting, sorting, filtering and zooming operations.

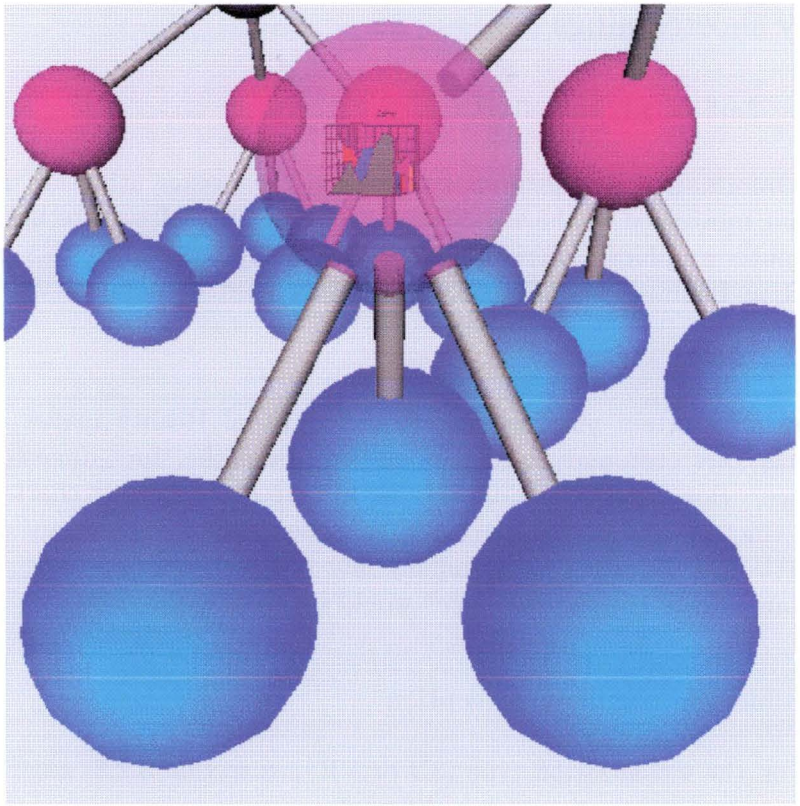


Figure 4.3: The transparency is used to inform the presence of an embedded chart.

For example, the transparency may allow to look through an object to see an embedded informational object.

4.1.5 Real-time and on-line visualization

Web-centric Virtual Reality technology, such as VRML, can deal with the need for real-time visualization. For example, it is critical in the fast paced nature of financial transactions where traders react to company, national and international news, at a moment notice. VR Business Information Visualization shows data in a format that improves comprehension, by dynamically representing fluctuations in data.

Dynamic, online visualizations the user can interact with by real-time manipulation can make the decision-making process easier.

4.2 Key shortcomings

4.2.1 The "in" and funny effect

3D and VR Information Visualization have received increasing attention in the last few years, motivated by the advances in hardware and software technology for 3D computer graphics. Therefore, more than really responding to a need, 3D and VR in Business Visualization are mainly used because they are in fashion.

Furthermore, "Entertain your audience" and "Virtual Reality makes compelling charts" are customary selling points to praise the three-dimensional user interface. However decision-makers do not need compelling charts but usable and efficient charts. We may also suppose decision-makers are using charts to settle the future of the company and not to "entertain their audience". "Communicate to your audience" would be more appropriate.

4.2.2 Problem in cognitive perception

Depending on the visualization type (3D or VR, see section 3.2.3, page 33) , the problem in cognitive perception is different.

4.2.2.1 3D Charts

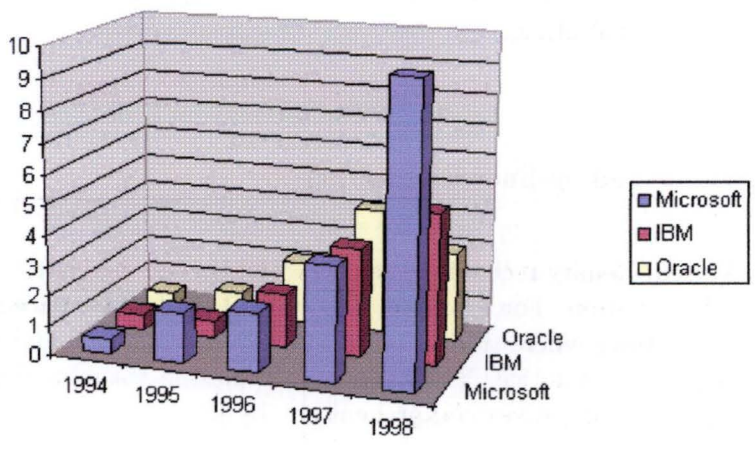


Figure 4.4: A three-dimensional Column Chart

After a comparison we have made between three-dimensional charts bundled in market softwares, we noticed that a lot of them are often adapted from 2D.

However 3D charts seem often more complex to interpret in comparison with their 2D counterparts. Charts from pictures 4.4 and 4.5 were both generated by MS Excel from the same data set. The 3D column chart is visibly confusing and less clear than the 2D histogram. Firstly, it is extremely tricky to read the correct values of columns because of the problem of perspective. Secondly, some columns are hidding other ones.
The problem of perspective view is particularly visible in figure 4.6.

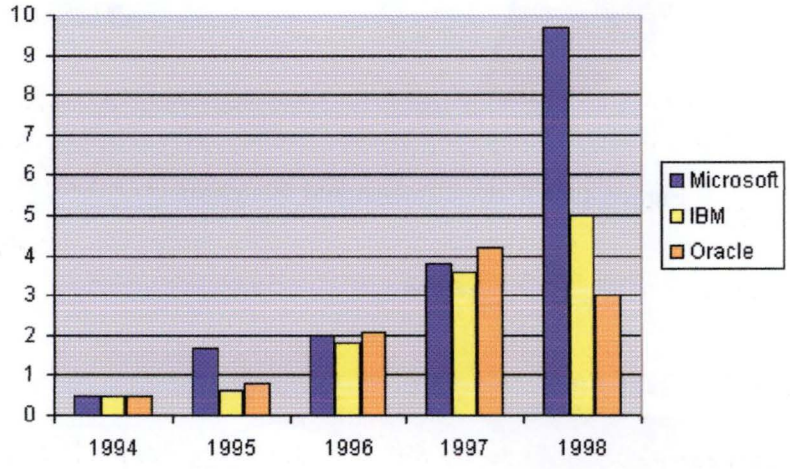


Figure 4.5: A two-dimensional histogram

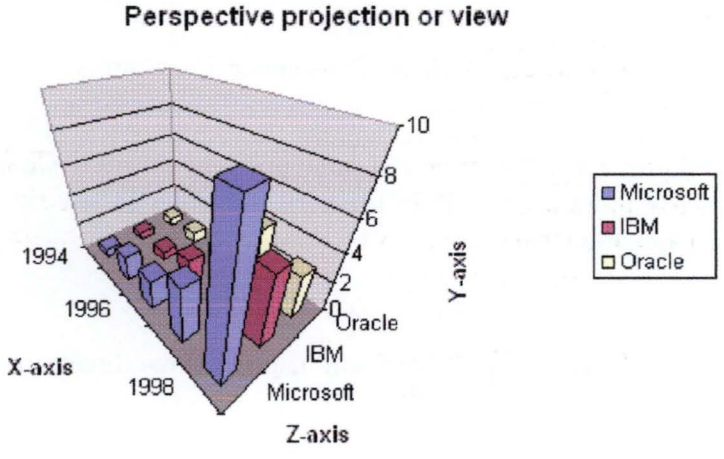


Figure 4.6: A perspective view

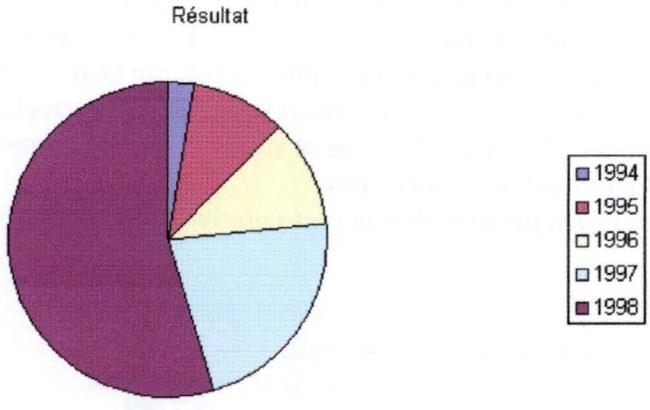


Figure 4.7: A two-dimensional Pie chart

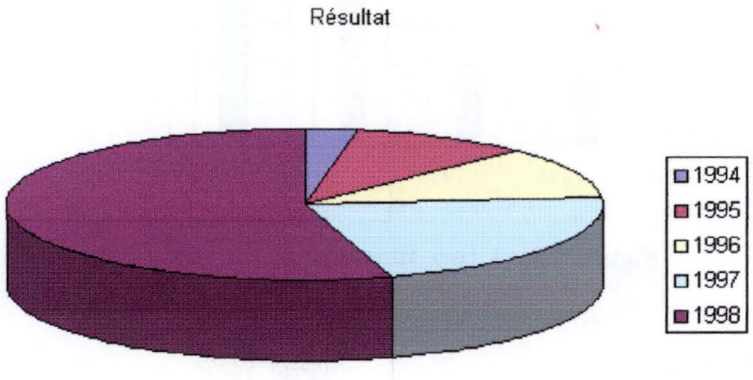


Figure 4.8: A three-dimensional Pie chart

Pie Chart in figure 4.8 suffers from the same problem since Pie Chart in figure 4.7 seems clearly more usable. Does the cosmetic effect justify the use of the three-dimensional Pie Chart ?

This question leads us to a new one: why using a three-dimensional look for charts existing in two-dimensions ?

Therefore, we met again the issue of the cognitive perception. If 3D selling point is more based on the cosmetic effect than on its efficiency, this certainly questions the usability and the interest of these new look charts.

4.2.2.2 VR Charts

We have just shown that three-dimensional charts adapted from two-dimensional ones do not seem convincing. What do usability and efficiency of VR charts become knowing that the Virtual Reality technology is based on the three-dimensional model ?

Benefits of Virtual Reality lessen the negative impact of three-dimensional visualization. However, as already touched on in the section 4.1.3, interactivity and manipulations demand an outstanding interface in order to be usable.

4.3 The made-to-measure VR Charts

In our opinion, Virtual Reality will be able to deliver all its potential **only with charts specifically designed to exploit its virtual nature**, as for instance the Infoball or Giotto3D. The creation of a whole new generation of charts purposely designed to make the most of the Virtual Reality is subsequently necessary. But this involves two major concerns.

1. The first one, as mentioned in section 2.4 (page 26), is the efficiency of the final visual representations for human vision. Since we are used to 2D visualization, we suffer from a lack of culture and training in 3D abstract representation. Its effectiveness has to do with human visual perception characteristics and cognitive processes of visual information processing. The problem of perspective illustrates this. Human depth perception is not as powerful as our ability to compare 2D spatial positions [Chuah-Eick98], and this can sometimes have an effect on human ability to interpret the graph accurately.

   Weber's law confirms this as [Ricard99] explains:

   > "Weber's law states, roughly, that our ability to detect a difference between two objects with a certain attribute (such as the difference between the length of two line segments) is related to the percent difference in the attribute, not the absolute difference. Weber's law tells us that the accuracy of length perception depends on the relative difference - not the absolute size of the difference. Weber's law applies to attributes other than length for instance, area and position."

   One consequence of this law is that three-dimensional charts are good **for depicting the general nature of data** and are almost never used to convey specific values, since it is difficult or impossible to determine exact values from such charts.

2. As already mentioned, the interactive nature of Virtual Reality is also a matter of concern. Interactions such as walking, studying, panning require beforehand a training. For instance, free rotation (i.e. in any directions) can cause disorientation to the user and confuses him. Permitting rotations only along one or two specific axes could be a good way to eliminate this problem. Moreover, thorough analyses of VR charts require too much manipulations and reduce the usability of the Visual User Interface. This underlines once more the necessity of a usable interface.

4.4 Conclusion

As a rule, three-dimensional charts are good for depicting the general nature of data and are almost never used to convey specific values, since it is difficult or impossible to determine exact values from such charts. However, if the reading of exact values is necessary, those are many times noted on the chart or a reference table may be provided.

In conclusion, the Virtual Reality technology has a future in complex problem-solving and decision-making. The Visual User Interface, the best cost-effectiveness of the screen space, the various visual cues and the real-time visualization are the keys of the Virtual Reality power which is able to convey information very efficiently as well as to provide interactive capabilities to exploit the maximum of business data.

However, analyses using VR are new in business environments and many researches have to be led in order to settle human visual perception problems and the interface usability.

Other interesting researches are led about information visualization, notably information-rich glyphs for software management data [Chuah-Eick98] and the remarkable spreadsheet paradigm to explore large and complex data sets [Huai-Hsin98]. The future will tell us if VR Business Information Visualization will be able to deal with these new techniques !

Part II

Research on the Design Process Requirements for the Development of a VR BIV Application

Introduction

The first part of the thesis has presented an analysis of the Virtual Reality usefulness in the Business Information Visualization field.

Now, we are going to determine the general design process requirements for the development of a VR BIV application. The development of a VR BIV application is a part of the global process for the visualization of business data.

According to us, the global process for the visualization of business data could be structured into these four stages :

1. the design process of a VR BIV application
2. the development of a made-to-measure application complying with the design process requirements
3. the creation of a chart thanks to the made-to-measure application
4. the chart visualization thanks to the made-to-measure application

A short description of each stage is provided right below.

The design process of a VR BIV application

We conclude in chapter 4 that the Virtual Reality technology has a future in complex problem-solving and decision-making. Many key features of the Virtual Reality are able to convey information very efficiently as well as to provide interactive capabilities to exploit the maximum of business data. That is the reason why we find important to determine in this second part what could be the general requirements of the design process of a VR BIV application.

The design process of a VR BIV application is essential for the efficient development of a usefulness and fully functional application. According to us, a fully functional application must be constituted of at least the following *four components*:

1. a visualization environment
2. a navigation tool
3. a graphical user interface (GUI) to perform real-time modifications and dynamic queries
4. a High-Level API[2] for dynamic generation of charts and behaviours programming facilities

In order to discover the general requirements of the design process, we aimed to develop by ourselves a VR BIV application. We define three general requirements, each one being discussed in one of the following chapter.

Chapter 5 intends to introduce the three-dimensional symbolic objects which are required for the development of a VR BIV application. For each of them, we shall next explain their use in the representation of business data.

Chapter 6 will present the technological requirements that the implementation platform must fulfill to allow an efficient development of *the four components*. Furthermore, in order to develop our own application, we shall check the requirements compliance of two selected platforms in chapter 6: VRML97 and Java3D.

Chapter 7 will focus on the fourth component, since we have designed such an API during our training at the University of Port Elizabeth[3] under the supervision of Professor Janet L. Wesson and Senior Lecturer Leon Nicholls.

Finally, in chapter 8, we shall check the validity of the requirements of our design process. With this in view, we shall present a little VR BIV application that we have programmed.

The development of a made-to-measure application complying with the design process requirements

The design and the development of a made-to-measure application is required, since the decision maker will be unable to create charts without it.

[2]**Application Programming Interface**

[3]**Training at University of Port Elizabeth (UPE), South Africa, from September 1999 to January 2000.**

As any design project, the application must be useful. The usefulness of an application is determined by two concepts: ***utility*** and ***usability*** [Nielsen98].

- *Utility*: does the system do anything that people care about ? If the system does something irrelevant or if it does not solve the main problem, then it does not matter whether it is easy to use: it will be a poor system in any case.
- *Usability*: can the user use the system and can he or she do so effectively ? Even if the system does exactly the right thing in theory, it will still be a poor system if the user cannot figure out how to get it work.

This stage is not discussed in our thesis. However, we shall often refer it to discuss the other stages.

The creation of a chart thanks to the made-to-measure application

The creation of a chart consists in

1. the chart **selection**:

   The selection of the appropriate chart was slightly overviewed in chapter 3 . It is function of the semantic properties of business data (see section 3.3).

2. the chart **instanciation**:

   The chart instanciation is the parametrization of the *symbolic objects*[4] properties by means of the application. The symbolic objects properties are instanciated according to business data values.

3. the chart **modelling**:

   The chart modelling consists in the conversion of the instanciated objects symbolizing the chart in three-dimensional models.

This stage will also often be refered to discuss the first stage.

The chart visualization thanks to the made-to-measure application

The visualization consists in the display of a chart within a virtual environment. A chart is the visual representation of a *symbolic chart object* designed with the implementation platform and instanciated by the application.

[4]As discussed in chapter 5, symbolic objects are the *primitive objects* and the *additional components*

We focused on the visualization stage in the first part of our thesis. Remind us that the Visual User Interface, the best cost-effectiveness of the screen space and the various visual cues are some of the keys of the Virtual Reality power which is able to convey information very efficiently as well as to provide interactive capabilities to exploit the maximum of business data.

Chapter 5

Chart Modelling in 3D

5.1 Introduction

In this chapter, we shall introduce the most common three-dimensional symbolic objects which are required for the development of a VR BIV application. For each of them, we shall explain their use in the representation of abstract data.

We distinguish three kinds of symbolic objects : the *primitive objects*, the *additional components*, and the *charts objects*. The *primitive objects* (see section 5.2) and the *additional components* (see section 5.3) are composing the *charts objects* which are designed to be converted into visual representations.

The chapter will first introduce the *primitive symbolic objects*.

5.2 Primitive symbolic objects

Basically, we call a *primitive object* the elementary component used in the building of charts. A primitive object is principally constituted of two types of features : the *geometry* (see section 5.2.1) and the *appearance* (see section 5.2.2). Moreover, some additional features may be included in order to clarify the primitive semantics.

5.2.1 The geometry

The geometry of a *primitive object* is its *geometric shape* (which gives its name to the object itself) and its *dimensions*. For instance, a cylinder is defined by its height and its radius.

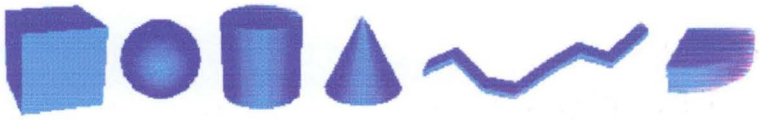


Figure 5.1: Some primitives

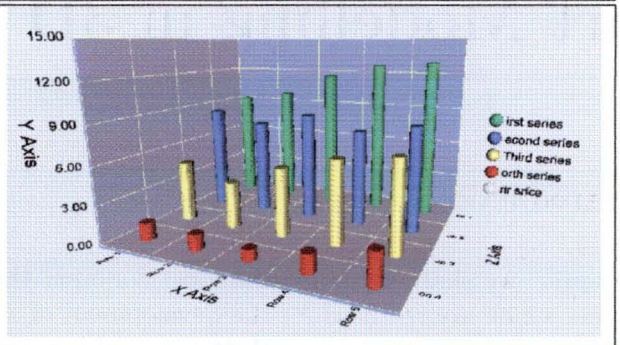
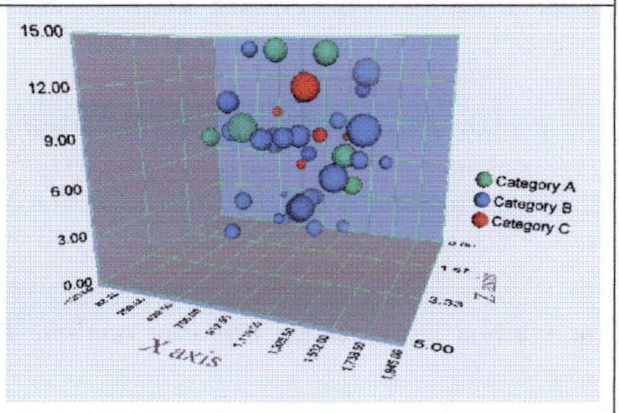
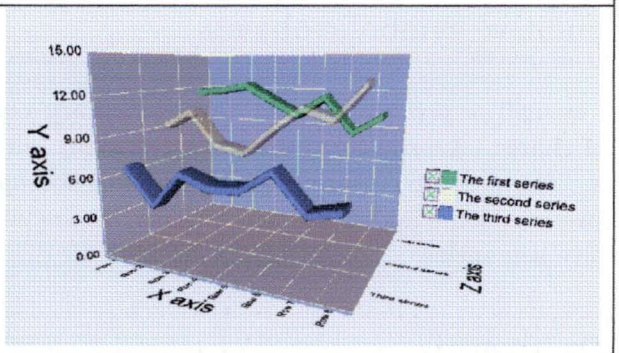
Primitives	Examples of corresponding charts
	
	
	

Table 5.1: Primitives and examples of corresponding charts

Figure 5.1 shows some primitives while table 5.1 illustrates their use.

1. **Geometric shape**

   We already introduced in section 4.1.4 (page 68) that *geometric shape*, dimension, position, colour, motion and other attributes are the signs of reality, which convey its information most efficiently. Therefore, primitive objects of various *geometric shapes* may be used in the same chart to represent different data dimensions or highlight an outlying information.

   For example, quarterly sales may be represented by cylinders while sales average may be represented by cones (see figure 5.2).

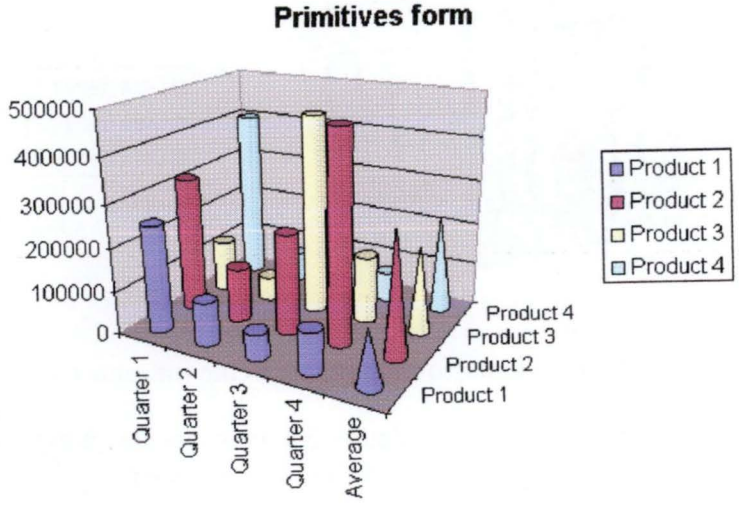


Figure 5.2: The use of various primitive geometric shapes inside a chart

2. **Dimensions**

   Each *dimension* of a primitive may be used to represent a data dimension (see section 4.1.4). It can be easier for the decision maker to compare heights of cylinders and therefore compare the corresponding data values.

5.2.2 The appearance

The appearance is determined by visual cues. The visual cues of three-dimensional objects may be used in VR charts to represent data dimensions or simply to clarify the charts. Some of those features have been mentioned in the first part (see section 4.1.4). We have picked out six visual cues: four of them belong to the primitive objects whereas the two left clarify the primitive objects semantics.

1. **Colour**

   Colour is used :

   - to express data dimensions
   - to underline specific information
   - to help to distinguish data series

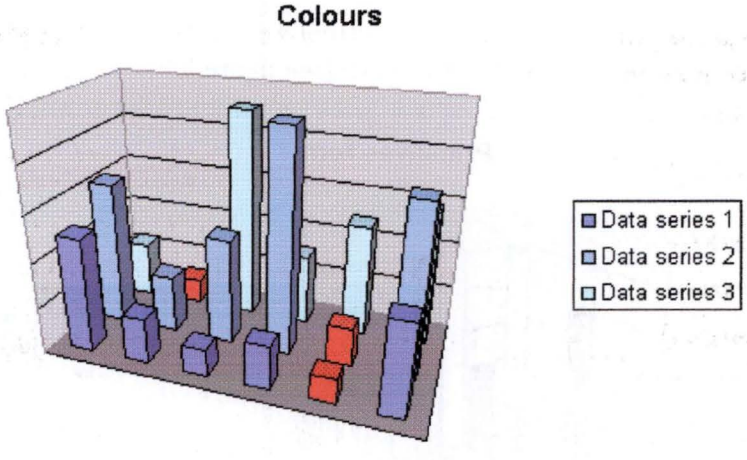


Figure 5.3: Use of colours to underline specific information

For instance, the red colour in figure 5.3 may be associated by the decision-maker easily with the idea of failure or problem.

1. **Texture**

   A *texture* may be a picture, a text or a set of colours applied to a primitive. This feature may simply be used to point out information as well as structure and clarify them (see figure 5.4).

2. **Transparency**

   The *transparency* is used :

   - to show the percentage of certainty of a given information
   - to show the importance of the data (a more transparent shape corresponding to a relatively unimportant information)
   - to allow to look through an object to see the embedded informational object (see figure 4.3, page 69).

The transparency of the right-handed edge on figure 5.5 shows the 30 % certainty of the information provided by the Child Node whereas the left-handed edge shows a certainty of 100%.

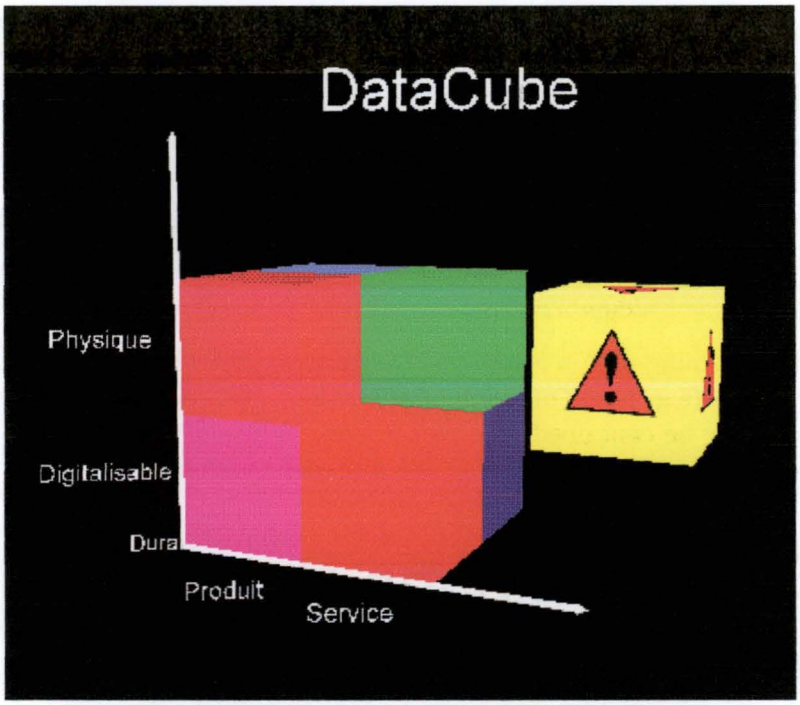


Figure 5.4: A texture is drawing attention on the *physical lasting service*

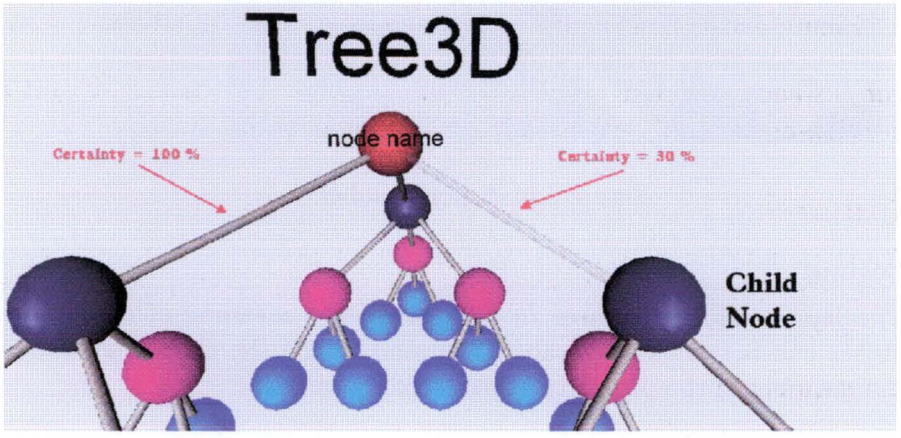


Figure 5.5: Percentage of certainty of a child node

3. **Glow**

[AlterVue98] defines *glow* as

> "the characteristic of colour saturation. If the colour has a greater saturation, then it is deeper colour, while less saturation will be a pale version of the same colour."

An appropriate means to use the glow effect is by animating a passage between two saturations of a colour, resulting in a *"flickering effect"*. Flickering can help to highlight a piece of information within a complex chart.

Here follows the two features clarifying the primitive semantics:

4. **Label**

   The *label* coming with a primitive is a comment providing any additional information (e.g. name or value of a variable). This text may always be displayed or be contained in a *popup label.* A *popup label* appears only while the mouse cursor is flying over a predefined zone.

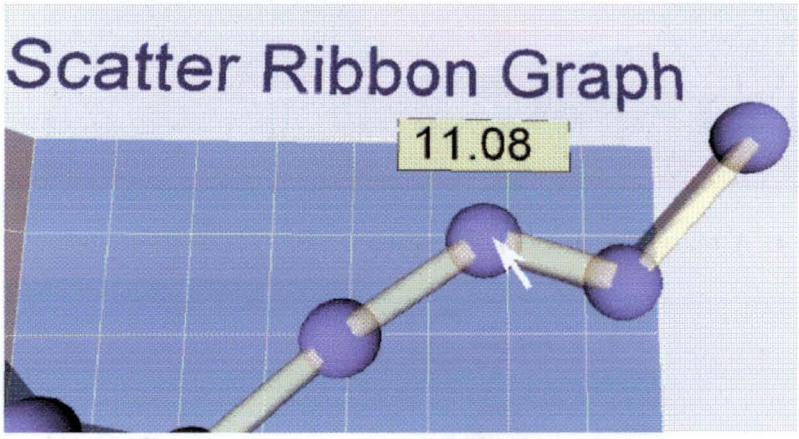


Figure 5.6: A popup label

5. **Legend**

   The *legend* allows the user to understand the meaning of a primitive object cue better. For instance, a legend beside the chart may expose the meaning of the primitive objects colours.

5.3 Additional components

A *VR Chart* consists of a logical assembly of primitive objects (see section 5.2, page 81) resulting in a meaningful structure. Remind us that the purpose of a chart is helping to make some sensible decisions.

However, the primitive objects are only the *"chart kernel"*. In order to make the chart readable, usable and efficient, additional *non-data* components are required.

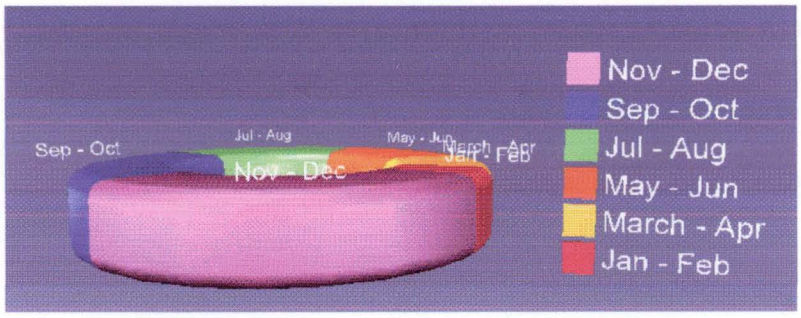


Figure 5.7: A Donut Chart and its legend

1. **Axes**

   The *axes* are the common means to determine data dimensions values as well as express reference marks[1].

   Figure 5.8 presents the annual results of three companies. 1994, 1995, 1996, 1997 and 1998 are the various values of the **"year"** dimension. A column is located according to a society name and a given year.

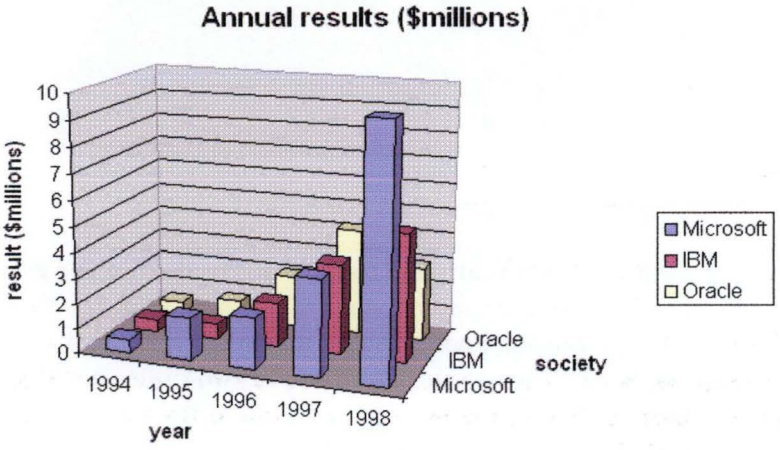


Figure 5.8: A chart with labelled axes.

2. **Label**

   The *label* is any information attached to the chart in text form. This information can be the chart title, the captions or the scales of axes.

   This feature clarifies the chart and guides the user in the comprehension of the information. Popup labels could be used to avoid the text overloading of the chart. Figure 5.4 (page 85)shows a title and the different

[1] In order to locate primitives by means of its coordinates

values of each axis. Figure 5.9 shows y-axis scale and the different values of any dimension on the x-axis.

3. **Grid**

   The *grid* is the part of the chart constituted of two cross-ruled walls and a floor. Joined with the axes and the labels, the grid helps to determine the data dimensions values represented by the primitive objects (see picture 5.9)

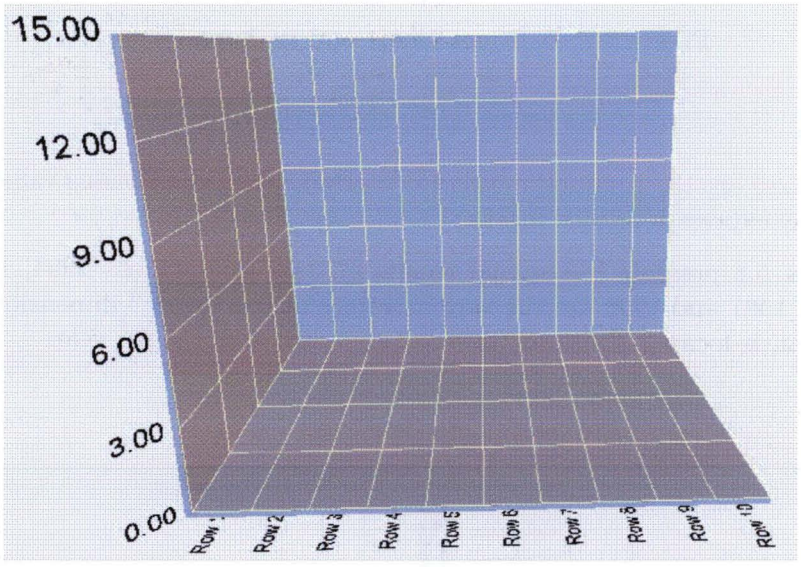


Figure 5.9: A grid with cross-ruled walls

Projecting additional information onto the walls or floor of the graph can also provide assistance in its interpretation. For instance, the first figure of table 5.2 shows contour lines of three-dimensional surface chart projected onto the floor. The second figure shows two-dimensional columns (projected onto the walls) representing additional data .

4. **Predefined viewpoints**

   Predefined viewpoints are "camera" placed in the space from which the user looks at the scene. Multiple viewpoints may help the user in a simple efficient way by adding a touring effect. It is specially interesting to study voluminous charts or to visit a World Chart[2] easily.

5. **Background**

   The *Background* is used to specify a colour backdrop that simulates *ground* and *sky*, as well as a background texture, or *panorama*. The background is placed behind all charts in the scene [Carrey-Bell97]. The

[2]see section 3.4.1.11, page 56

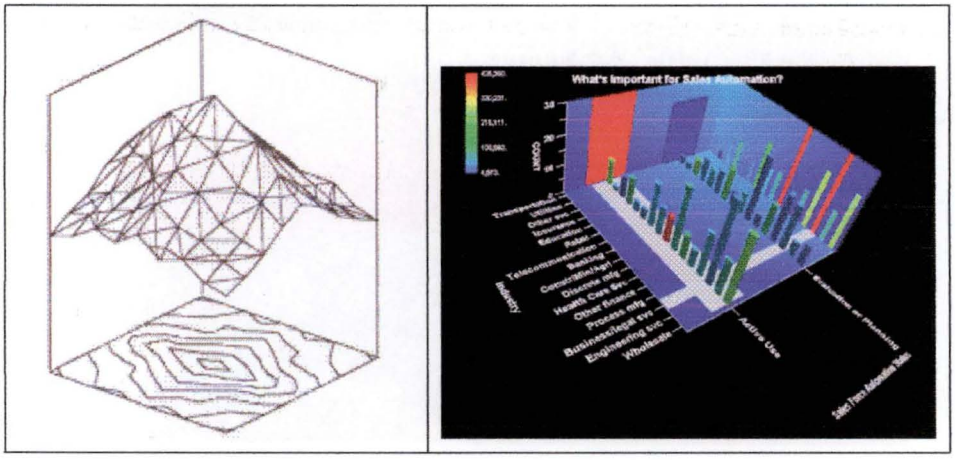


Table 5.2: Two projections type

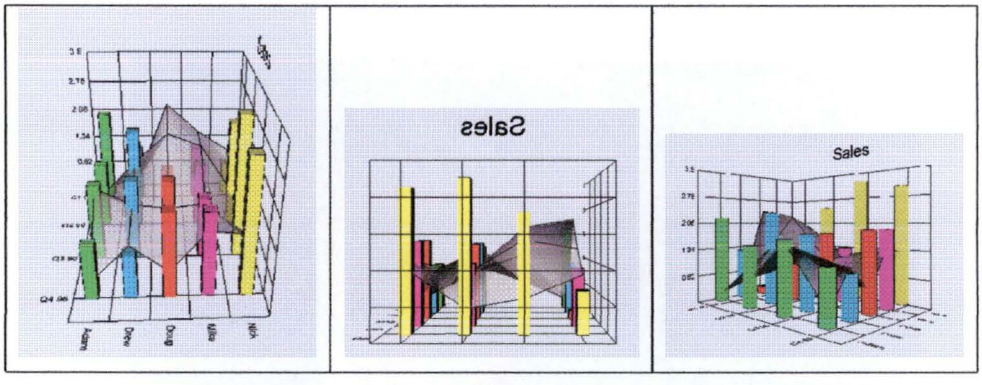


Table 5.3: Three viewpoints of a Column Surface Chart

figure 5.10 shows the information contained in the red cube of the figure 5.4 (page 85). The **red** background points out that we are standing inside the **red** cube.

6. **Light**

   A *light* allows to highlight a specific area of the chart.

7. **Drill-down**

   The *drill-down* technique may allow to reach a more granular level of data from the current chart, resulting in a treelike hierarchy of charts. For example, figure 5.12 shows an embedded chart inside a node.

8. **Animation**

   Animation can be used to show evolution of the same chart along time periods. Time is represented by storing various phases of a chart and then animating them (see section 3.4.1.10, page 55).

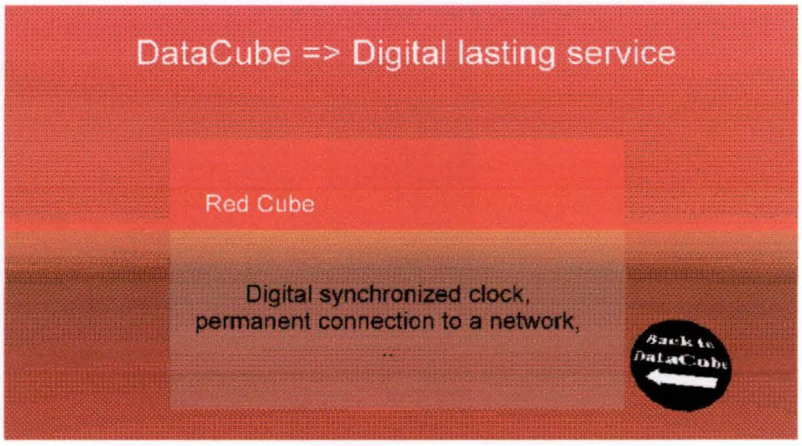


Figure 5.10: A red background

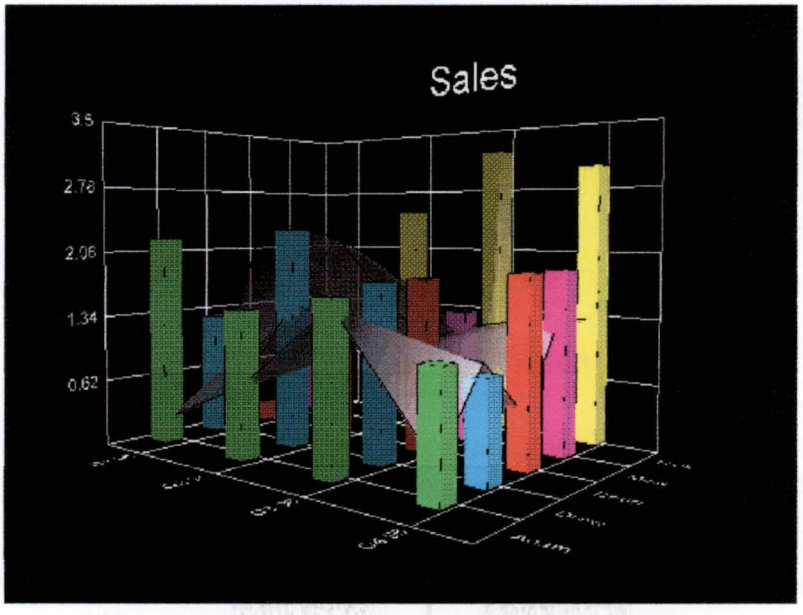


Figure 5.11: A spot is lighting a zone of the chart.

9. Manipulation of specific objects

The *manipulation of specific objects* of the chart may be very useful to analyse them independently of the whole chart. For instance, all the areas of an Area Chart are often not visible directly. That is why the user gets the possibility to pick up specific areas (see figure 5.13).

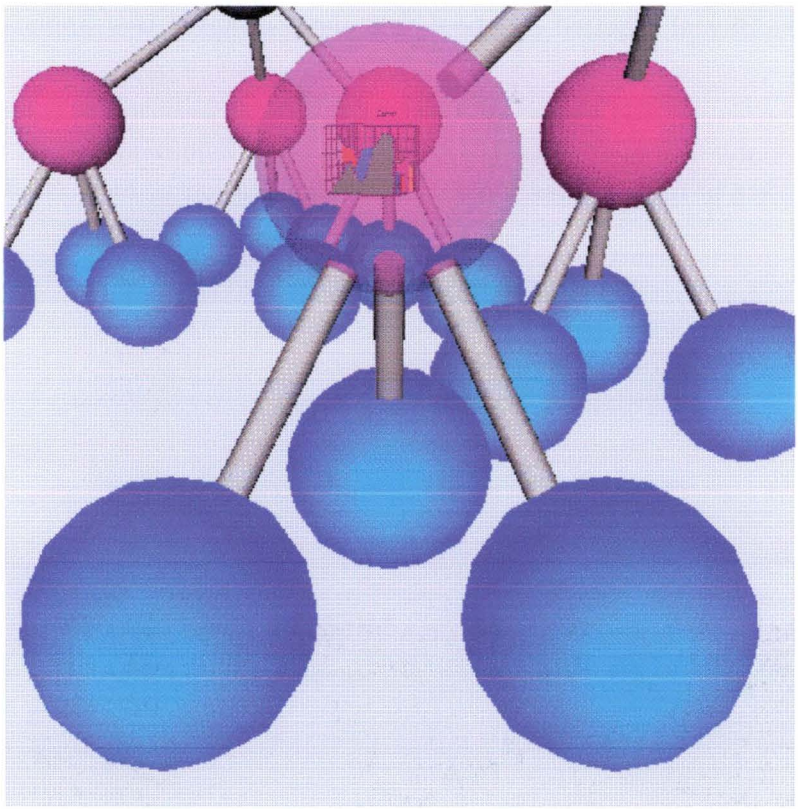


Figure 5.12: An embedded chart.

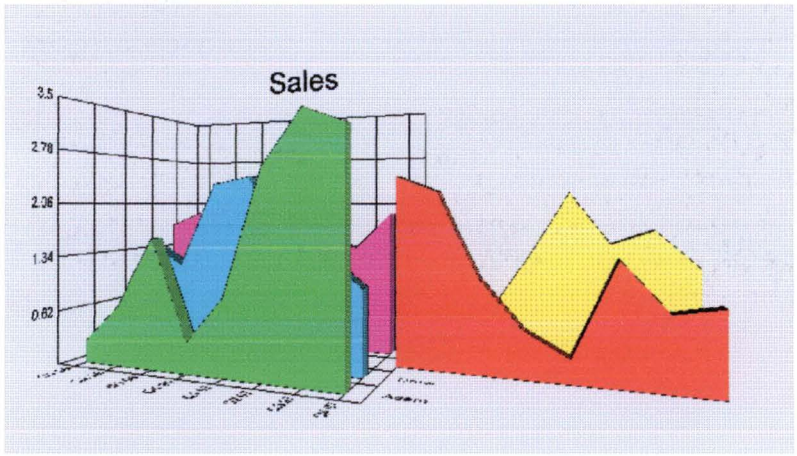


Figure 5.13: Some areas are dragged out of the chart

5.4 Conclusion

Chapter 5 introduced three-dimensional symbolic objects (i.e. the symbolic primitive objects and all the non-data additional components making the sym-

bolic charts objects) required for the development of a VR BIV application.

The next chapter will now present both technological requirements and technological choice in order to develop a VR BIV application as most efficiently as possible.

Chapter 6

Technological Requirements & Technological Choice for BIV

6.1 Introduction

According to us, the implementation platform must allow to develop - or provide - at least four components so as to program a fully functional VR BIV application:

1. a visualization environment
2. a navigation tool
3. a graphical user interface (GUI) to perform real-time modifications and dynamic queries
4. a High-Level API for dynamic generation of charts and behaviours programming facilities

In order to develop these four components as most efficiently as possible, we are now presenting the technological requirements which seem important to us and that the technology chosen for the implementation platform of the four components must fulfill. Furthermore, in order to develop our own application, chapter 6 will check the requirements compliance of two technologies featuring three-dimensional capabilities. This analysis should lead to the best informed choice for the implementation platform of our own VR BIV application.

Among the various available technologies evolving around 3D, we decided to select VRML97 and Java3D according to two criteria: *capabilities* and *universality*.

- The programming of three-dimensional applications requires two major *capabilities* of which relative importance depends on the developer aims. These are the *modelling of three-dimensional contents* (allowing to visualize a virtual world) and *a runtime system* (allowing to modify as well as to manipulate the virtual world dynamically).
  We selected VRML97 and Java3D because of their respective key capability. VRML97 is predominantly a file format for three-dimensional contents while Java3D is predominantly a 3D graphics runtime system. Therefore, we shall have to determine how good each technology is regarding its key capability and how good it is to accomplish the one of its competitor. Our choice will depend on the balance of their respective qualities according to our criteria.

- Regarding *universality*, VRML97 is the standard file format to exchange three-dimensional contents on the Web and using it means potentially reaching over 50 million people. Java3D is a part of the JavaMedia suite of APIs, making it available on wide range of platforms. It also integrates perfectly with the Internet since applications and applets written with the Java3D API get access to the entire set of Java classes.

To support our conclusions, Appendix A proposes a technical analysis of the architectures of VRML97 and Java3D while Appendix B presents two versions of the same program, written respectively with both technologies.
Appendix C proposes an example of a program written in Java which displays in real-time three-dimensional VRML contents.

6.2 Technological requirements

We have specified a list of technological requirements that the technology chosen for the implementation platform of the four components must fulfill. These technological requirements should allow to develop a fully functional, optimized and usable VR BIV application and this, in the most convenient development way.

We shall present the requirements according to their importance in the final decision we will make.

1. **Dynamic charts building and behaviour programming**

   It is of course essential for the technology to allow the dynamic building of charts as well as the programmation of some behaviours[1] (see chapter 5). The dynamic building of charts is obvious when real-time visualization

[1] e.g. a glowing effect, popup label, touring effect, drill-down, animation, and so forth

and dynamic queries will imply the frequent rebuilding of new charts on-the-fly (see section 4.1.5, page 69).

2. **Visualization**

   It is as well essential to allow the display of three-dimensional contents.

3. **Navigation**

   Navigation is the feature required by the Visual User Interface and the interactivity (see section 4.1.2, page 66) as well as the best-cost effectiveness of the screen space (see section 4.1.3, page 68).

4. **Availability of basic elements**

   The technology must define most of the basic elements found in today's 3D applications such as hierarchical transformations, light sources, viewpoints, geometry, animation, material properties, and texture mapping. These features are required since they contribute to the technology universality (see point 8) as well as they enchance data visualization (see section 4.1.4, page 68).

5. **Ease of development**

   From a developer point of view, we prefer the most convenient way to develop the application.

   (a) simplicity

   A *simple* language to describe three-dimensional contents and specify behaviours will make our task easier.

   (b) reusability

   The *reusability* is a crucial feature for the ease of development. This will prevent us from reinventing the wheel. As a matter of fact, a library of primitive objects[2], provided by the language or programmed by ourselves, will make the chart creation easier.

6. **Usable Graphical User Interface**

   When designing an application, one of the most important point to focus on is its *usability*. This concept of usability has been fully discussed by Jakob Nielsen [Nielsen98], which defines it as follows:

   > "*Usability* is the measure of the quality of the user experience when interacting with something - whether a web site, a traditional software application, or any other device the user can operate in some way or another"

[2]see section 5.2

As suggested in sections 4.2.2.2 and 4.3, the interactivity and manipulations demand an outstanding interface in order to be usable. Therefore, the technology must give the opportunity to the programmer to design a **usable** navigation tool and a **usable** Graphical User Interface[3].

Usability features five criteria [Nielsen98]:

(a) **ease of learning**: How fast can a user, who has never seen the user interface before, learn it sufficiently well to accomplish basic tasks ?

(b) **efficiency of use**: Once an experienced user has learned to use the system, how fast can he or she accomplish tasks ?

(c) **memorability**: If a user has used the system at some earlier date, can s/he remember enough to use it more effectively the next time (or does the user have to start over again learning everything every time) ?

(d) **error frequency and severity**: How often do users make errors while using the system, how serious are these errors and how easy is it to recover from a user error ?

(e) **subjective satisfaction**: How much does the user like using the system ?

The all five criteria need to be considered in any designed project but their importance depends on the project itself and on its operating environment. Section 4.1.3 told us that the great flexibility of Virtual Reality depends on particularly well designed interfaces. We have performed several trials of navigation tools and we have felt intuitively that the critical user criteria in this context were the *ease of learning*, the *efficiency of use* and the *subjective satisfaction*. However, as it is always a challenge to combine the *ease of learning* and the *efficiency of use*, the developer must try to reach the best balance between both. Regarding the *subjective satisfaction*, we consider that the decision-maker should be naturally more satisfied using a VR BIV application rather than using its usual one.

7. **Performance**

(a) optimized implementation

The rendering[4] of a virtual world is a task that typically demands plenty of resources to a computer. The technology must feature an *optimized implementation* making the charts rendering as light as

[3]two of the four components of a fully functional VR BIV application, according to our specifications.

[4]i.e. the calculation by the computer of a three-dimensional content

possible. This requirement is obvious when real-time visualization and dynamic queries will imply the frequent rebuilding of new charts on-the-fly (see section 4.1.5, page 69).

(b) scalability

The technology must be *scalable* enough to allow the building and the manipulation of small charts as well as complex charts (as voluminous World Chart, complex drill-down hierarchy and so forth)

8. **Universality**

The *universality* of the technology assures that the application may be run by any computer on any platform and that the technology file format will be adopted as a file interchange format. We may distinguish three criteria to assure the technology universality:

(a) file interchange format

As a file interchange format, the technology may gain a widespread support.

(b) cross-platform

The asset of a cross-platform application is obvious. One application implementation runs on every platform. It is a gain of time and money.

(c) network-centric

During an application use, the data could need to be updated with data conveyed by a network (see section 4.1.5).

6.3 VRML97

6.3.1 Overview

VRML is an acronym for the Virtual Reality Modelling Language. It is a web-centric technology which allows the user to describe interactive three-dimensional worlds.
These virtual worlds, downloaded from the Internet, are interpreted and rendered in real time by a VRML browser. As a matter of fact, the interpretation, execution, and presentation of VRML files will typically be undertaken by a mechanism known as a *browser*. Usually, the browser provides a navigation tool to move within the world.
It is a truly open standard (ISO/IEC 14772-1:1997), specified by the VRML Consortium[5], hence not owned by any particular company [Carrey-Bell97].

[5]http://www.vrml.org/home.html

VRML defines most of the commonly used elements found in today's 3D applications such as hierarchical transformations, light sources, viewpoints, geometry, and so on.
One of the primary goals in designing VRML was to ensure that it at least succeeded as an effective 3D file interchange format. It allows integration of three dimensions, two dimensions, text, and multimedia into a **coherent** model. We may think of it as the 3D equivalent of HTML.

VRML is not a programming language but a descriptive language.

A programming language typically defines an Application Programming Interface (API) and involves code compilation, libraries linking and a restrictive syntax. That is why a descriptive language is more simple and intuitive than a real programming language.
The price to pay for this ease-of-use is some limitations of VRML, but they often can be overcome, mainly with the *Internal Script Node* and the *External Authoring Interface* (EAI) techniques. Both ISN and EAI are scripting language integrated within VRML which enables rich interactions between external processes[6] and the virtual world. Furthermore, those scripting languages allow to program some behaviours and more complex effects.

At the present time, the last release of VRML is VRML97, which is a revised version of VRML2.0 made in collaboration with the Java3D team to improve the interaction between the two technologies.

6.3.2 Requirements compliance

1. Visualization

   VRML is a descriptive language designed to model virtual worlds and therefore, renders three-dimensional spaces appearing on a display screen. Consequently , VRML allows of course the visualization of three-dimensional contents.

2. Navigation

   As introduced in the overview, a VRML browser provides a navigation tool to move within the world.

3. Availability of basic elements

   VRML allows the developer to give a world a huge set of visual effects (colour, texture, fog, lighting) and behaviours (Animation, Interpolators[7]). Thanks to interpolators, the programmer is able to add to his

[6] such as a Java applet
[7] Interpolators let also add animation to virtual world

world some rotation and scale effects, colour changes and movements along a defined path.

4. Dynamic charts building and behaviour programming

   VRML is a descriptive language, and at first sight, it is not designed to accomplish typical programming language tasks such as dynamic charts building. As introduced in the overview, this limitation may be overcome with the *Internal Script Node* (ISN) and the *External Authoring Interface* (EAI) techniques. The key idea is: "VRML is, ISN / EAI does"

   The Internal Script Node (ISN) and the External Authoring Interface (EAI) are the two specified methods to allow programmers to control the objects in the VRML world from within Java.

   The *Internal Script Node*(ISN), which belongs to VRML syntax, allows the programmer to control the content of a VRML browser window by affecting objects[8]. Furthermore, scripts allow the world creator to define arbitrary behaviours. As a matter of fact, VRML lets the programmer specify dynamic events that can affect primitive objects of the chart. VRML objects may generate events in response to environmental changes or user interactions. A behaviour possesses customizable sets of wakeup conditions which trigger the behaviour when satisfied. The VRML97 specification defines *Script* bindings for the Java and JavaScript languages.

   The *External Authoring Interface* (EAI) allows the programmer to control from a Java applet running in a web page[9] the content of a VRML browser window embedded on the same page. The EAI is a proposed Informative Annex to the VRML specification, i.e. browser implementations may choose to implement the EAI - or not - and still feature complete VRML97 compliance. Most browsers support Java, of course, as the primary language interface to program the EAI. However any language could theoretically be used to program to it.

   Therefore, the main difference between ISN and EAI consists of the location from which Java controls the virtual world. From *inside* regarding the ISN and from *outside* regarding the EAI. The selection between Script Node and External Interface depends on the specific project. Both solutions are very flexible and powerful. In most cases, **either can be used**. Script Nodes can be chosen for behaviours purely within the world and the External Interface for behaviours linking outside (for instance, if

[8] by updating some primitive objects attributes
[9] displayed by a WWW browser

the programmer wants to create integrated multi-media presentations, of which VRML is one type of media).

Some problems
It appears that some annoying technical limitations are inherent in EAI, whereas *Internal Script Node* works perfectly.

The first one is the serious instability of the environment. The implementation of EAI within VRML browsers is rather unstable and may cause uncertain results. Depending on the VRML browser used, EAI will not react the same way. This dramatically worsen the cross-platform capability of this technique.

Secondly, some browsers, notably the Cosmo Player browser, have difficulties with Java objects and memory. This means a memory error will crash the browser at some point. However it seems possible to eliminate this problem. It is raised in the VRML EAI FAQ[10]

5. Ease of development

   (a) Simplicity

   - *Simplicity* was an important VRML design constraint. The success of the HTML file format motivated this constraint since the HTML simplicity fueled the explosive growth of the World Wide Web (for instance, it is possible to create Web pages with any generic text editor and it is relatively easy to design an HTML browser). A simple specification encourages the development of more browsers, which leads to competition, which will encourage the development of higher quality implementations.
     Despite this simplicity, a complex universe requires a good control of the language [Carrey-Bell97].
   - VRML is a *content* language that describes interactive three-dimensional worlds. The syntax is more intuitive and more accessible than a programming language.
   - In order to gain time and ease of use, the programmer can design three-dimensional contents with a specialized VRML editor (VRCreator 2.0 from Platinum Technologies[11] or Cosmo Worlds from Cosmo software[12]). These editors handle much - and, typically, all - of the process of creating a basic world. The

[10] http://www.frontiernet.net/~imaging/eaifaq.html
[11] http://www.platinum.com
[12] http://www.cosmosoftware.com

editor adds the appropriate objects to the virtual world simply when the programmer uses "*drag and drop*" features from a list of prebuilt objects. In truth, the construction of a really useful world, only by using text editors, would be a very arduous task.

- Exploring a VRML world is very easy because many free browsers are available on the Net. The browsers are powerful and offer very interesting capabilities. The navigation tool[13] of Cosmo Player[14] (see table 6.1[15]) is an example.

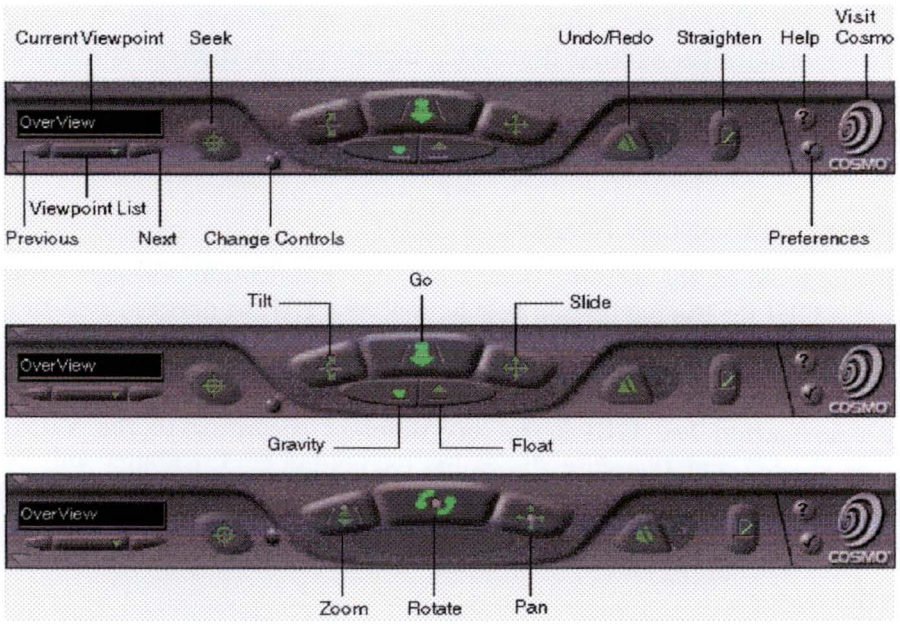


Table 6.1: The navigation tool of Cosmo Player

(b) Reusability

With simplicity, *reusability* is another property of a good file interchange format. It should be relatively easy to take files created by various people or tools and compose them together to create a new world. Reusability is notably allowed by *PROTO* and *EXTERNPROTO* statements.

PROTO statement

The *PROTO statement* aims to *encapsulate* and *reuse* a virtual world. It defines a new object type in terms of a combination of predefined object types. The prototyping can make VRML easier

[13] called a "dashboard"

[14] from Cosmo Software, http://cosmosoftware.com

[15] the pictures are coming from the Cosmo Player tutorial.

to use and can reduce the VRML size[16] of files.

Each prototype *instance* can be considered to be a complete copy of the prototype, with its own fields, events, and copy of the prototype definition. The definition of the PROTO objects must be stored internally, in the same file as the various instances.

EXTERNPROTO statement
The *EXTERNPROTO statement* features the same characteristics as PROTO except that the definition of the node type is stored externally (in another file). This interesting mechanism allows the definition of libraries of reusable objects, since we may design a file of EXTERNPROTO's and only reference the prototype we desire to use. Consequently, we could simulate the *package* concept of Java.

6. Usable Graphical User Interface

   As suggested in the introduction, a VR BIV application will contain at least a GUI for the navigation tool as well as a GUI to perform real-time modifications and dynamic queries.

   The *navigation tool* GUI looks very different according to the VRML browser used and therefore the global navigation tool usability will depend strongly on it. On the contrary, the *real-time modifications* GUI usability will depend on the final developer, and in the present case, us.

7. Performance

   (a) Optimized implementation

   - An important VRML design constraint was allowing *implementations* to be highly optimized. The performance ramifications of every feature were considered. For instance, if a fast implementation of something did not seem possible, the feature was rejected.
   - The normal extension for VRML files is "*.wrl*", though occasionally "*.wrz*" is used for compressed VRML files. As a matter of fact, browsers are able to read VRML files that are compressed with GZIP. Therefore large worlds in VRML remain bandwidth friendly.
   - A slow 3D graphics runtime is not usable. VRML is optimized for speed where possible.
   - In order to avoid useless rendering, the programmer can tell the browser which part of the shape it has to render. For instance, to avoid the drawing of a bottom side hidden by the floor.

[16] since the new object is defined once and once only.

- *Level Of Detail* (LOD) is an important optimization feature which
  - reduces "upfront" download time: the simpler models are loaded first
  - improves frame rate: simpler models are used for far away objects.

  When the viewer gets close of an object, LOD will then display higher-details models.

(b) Scalability

VRML97 is designed to scale in three ways :

i. Firstly, it should be theoretically possible for a VRML browser to handle a world distributed across the Internet that contains millions or billions of objects.

ii. Secondly, VRML97 should work well when used with both very powerful and very inexpensive machines, allowing the VRML97 browser to trade off image or simulation quality for improved performance and to scale well with increased hardware performance.

iii. Thirdly, VRML worlds should scale with network performance, from the 14.4K modems still in use today to multigigabit connections that might become common in the future.

8. Universality

(a) File interchange format

VRML 1.0 has gained widespread support as a file interchange format, and another constraint on the VRML97 design was to make it an even better interchange format.

Consequently, it was crucial for the new VRML release to offer the necessary objects to create rich, interesting and interactive worlds (i.e. shapes, textures, viewpoints, light effects, events, and so forth). Moreover, it should be easy for applications to both read and write VRML files.

One of the critical design decisions was whether it would be best to add animation and interaction capabilities to VRML97 by making it a full-fledged programming language. Programming languages are very powerful, but they make poor file interchange formats. One of the reasons HTML is so successful as a file interchange format is the wide variety of HTML editors that can read, modify, and write any HTML file [Carrey-Bell97].
The needs of both VRML browsers and VRML editors were considered in the VRML97 design.

(b) Cross-platform

The cross-platform capability is assured by the *file interchange format* feature of VRML. A VRML file is simply a text file that may be read on any platform and it is easy for applications to both read and write VRML files.

(c) Network-centric

VRML was designed to fit into the *existing infrastructure of the Internet* and the World Wide Web. It uses existing standards wherever possible, even if those standards have some shortcomings when used with VRML. Using existing standards instead of inventing new incompatible ones is much easier for the Web developer, who can use existing tools to create VRML contents. It also makes it much easier for somebody implementing the VRML standard, since libraries of code for popular standards already exist (besides, this improves reusability).

VRML files may contain references to files in many other standard formats. JPEG, PNG, GIF, and MPEG files may be used as texture maps on objects. WAV and MIDI files may be used to specify sound that is emitted in the world. Files containing Java or JavaScript code may be referenced and used to implement programmed behaviour for the objects in virtual worlds. Each of these is an independent standard, chosen to be used with VRML because of its widespread use on the Internet.

6.3.3 Conclusion

VRML97 meets our requirements list. VRML97 is the standard for rendering three-dimensional objects on the Web. VRML objects exist in a virtual world and can be linked to various objects such as Java applets, Javascripts, images, multimedia files and even URL's. Joined with Java and thanks to its skills and its simple architecture, VRML97 is a perfectly valid development platform for Business Information Visualization purposes.

If truth be told, many Business Visualization softwares, based on Virtual Reality, are using VRML as file format to visualize business data. Various examples are provided in the conclusion.

6.4 Java3D

6.4.1 Overview

The *Java3D API* is an Application Programming Interface (API) used to design three-dimensional graphics applications and applets. Java3D is a standard extension to the Java2 platform. The API provides a collection of high-level constructs in order to create and manipulate 3D geometry as well as structures to render this geometry. Java3D provides the functions for creation of imagery, visualizations, animations, and interactive 3D graphics application programs [Sun99d].

Besides, Java3D introduces some unusual concepts for graphics environment, such as 3D spatial sound.

The key advantages of Java3D are:

1. a **rich set of features** in order to create complex three-dimensional contents.
2. a **high level of performance** to application users thanks to meticulous design decisions.
3. a support for **runtime loaders**, allowing the accommodation of a wide variety of common file formats, such LightWave (.lwt), WaveFront (.obj), VRML (.wrl) and so forth.

6.4.2 Requirements compliance

1. Visualization

   As introduced in the overview, Java3D provides a collection of high-level constructs in order to create and manipulate 3D geometry as well as structures to **render** this geometry.

2. Navigation

   Java3D provides tools helping to the navigation through a world. Nevertheless, the programmer has to program his own navigation tool to walk around in the virtual world.

3. Availability of basic elements

   The API allows to give an object a huge set of visual effects (colour, texture, fog, lighting), behaviours (Animation & motion, Interpolators) and 3D spatial audio sound.

4. Dynamic charts building and behaviour programming

   Java3D programs may be written to run as stand alone applications or applets. The application capability does not exist in VRML unless using specific API as for instance Portal and VR DEV from EM7 Electrohouse [EM700]. Even ISN and EAI do not allow this since the display always occurs in a HTML page.

   Java3D includes the *Interpolator* class, which extends the *Behavior* class. *Interpolators* let add animation to Java3D programs. Thanks to *Interpolator* subclasses, the programmer is able to add to his program some rotations and scale effects, colour changes and movements along a defined path.

5. Ease of development

   (a) Simplicity

   - Java3D provides a high-level, object-oriented view of three-dimensional graphics. An application based on the Java3D API constructs individual graphics elements (as separate objects) and connects them together into a treelike structure called a *scene graph* (see appendix A). Using the *scene graph*, the Java3D environment removes much of the complexity of programming in 3D. Lower-level, procedural 3D APIs (like OpenGL[17]) need considerable 3D graphics experience. On the contrary, a scene graph is more intuitive and the rendering process is handled automatically by Java3D.
   - The creation of a world by defining every vertex of every triangle in every object is possible but it would be a very arduous task. In order to avoid to become crazy or waste too much time, the programmer will use the large and growing number of 3D loaders available to import content into the Java3D runtime. Some of these loaders load only the geometry, while others also load attributes and behaviours.
     Furthermore, Sun wrote a VRML loader to parse and translate VRML files and generate the appropriate Java3D objects and Java code necessary to support the files contents.
   - However, the Java3D architecture still remains more complex than which of a descriptive language. Besides, it is also too complex for our simple Business Visualization purposes (see Appendices A and B).

   (b) Reusability

[17] designed to obtain the best possible speed and give programmers the greatest possible control over the rendering process

An Oriented Object language like Java3D features *reusablity* as fundamental principle.

6. Usable Graphical User Interface

   The usability of both navigation tool GUI and real-time modifications GUI depends on the design made by the final developer.

7. Performance

   (a) optimized implementation

   - Geometry compression is programmer-definable. He specifies how much vertex resolution he is willing to give up, and Java3D can do the compression for him.
   - Many features specifically designed to increase performance were included in the API.
     The *renderer* automatically optimize in many ways for improved rendering performance. For instance, the Java3D API was designed with *multithreaded* environments features. At any moment, parallel threads are running performing various tasks such as visibility detection, rendering, behaviour scheduling, sound scheduling, collision detection, and so forth.
     Java3D uses *low-level, native libraries* as interface with the hardware. The reference implementations use either OpenGL or Direct3D, the most popular interfaces to graphics cards. This enables to get native hardware acceleration when manipulating 3D models.
   - Like in VRML, as the user gets closer to an object, Java3D can automatically substitute a higher-detail model.

   (b) Scalability

   Theoretically Java3D is able to run on everything from a laptop PC to a high end workstation , and to take advantage of the hardware as it scales. It also scales across a range of viewing environments (conventional screen to head-mounted display).

8. Universality

   (a) File interchange format

   Regarding Java3D, "application interchange format" would be more appropriate. As a matter of fact, Java3D does not define a file or network format of its own.
   Nevertheless Java3D is part of the *JavaMedia suite* of APIs, making it available on a wide range of platforms. It also integrates well with the Internet since applications and applets written with the Java3D API get access to the entire set of Java classes. Consequently, Java3D delivers Java's "*write once, run anywhere*" paradigm.

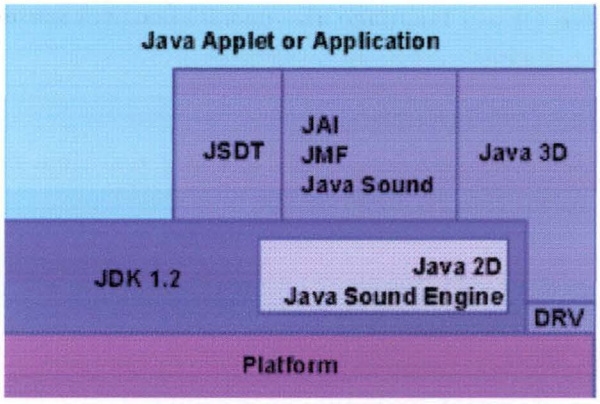


Figure 6.1: The JavaMedia Suite of APIs

(b) Cross-platform

The cross-platorm ability is a fundamental principle of Java.

(c) Network-centric

Java is a language designed for the network. Java3D extends that paradigm to 3D graphics, enabling collaboration in 3D across the network.

6.4.3 Conclusion

Java3D meets our requirements list. As Java3D belongs to the JavaMedia Suite, it inherits from all its advantages. It is able to create sharp visual effects and many behaviours as well as improve automatically the rendering in many ways. Therefore, Java3D is also a perfectly valid development platform for Business Information Visualization purposes.

However, this platform demands more development time than in VRML. Despite a whole bunch of features to make the Java3D API more accessible (scene graph, automatic rendering, and so on), Java3D keeps a very restricting syntax. Furthermore we should program our own navigation tool since powerful browsers, like in VRML, do not exist in Java3D.

6.5 Proposal of the implementation platform

6.5.1 Comparisons between VRML97 and Java3D

We have realized a deep analysis of the architecture of VRML97 and Java3D at UPE. The analysis result has been foremost significant to compare the

simplicity requirement of both technologies but it also gave us a more detailed knowledge of the capabilities of VRML97 and Java3D. This analysis is available in Appendix A.

6.5.1.1 VRML97 summary

The Internal Script Node is a very interesting technique which allows a full control over the VRML objects (section 6.3.2, point 4)

The VRML architecture is really simpler than a programming language one[18] (see Appendices A and B).

VRML is powerful enough to business data visualization purposes.

VRML is the standard interchange format file on the web.

6.5.1.2 Java3D summary

Java3D is a very powerful programming language. It is able to create sharp visual effects and many behaviours as well as improve automatically the rendering in many ways.

As Java3D belongs to the JavaMedia Suite, it inherits from all its advantages.

Despite a whole bunch of features to make the Java3D API more accessible (scene graph, automatic rendering, and so on), Java3D keeps a very restricting syntax. Too many code lines are required to create simple shapes and to change simple features of the virtual universe, unlike VRML.
As a matter of fact, Java3D is certainly easier to use than low-level libraries but remains too heavy for Business Information Visualization purposes. At this moment, Java3D targets only 3D professionals.

We should program our own navigation tool since powerful browsers, like in VRML, do not exist yet in Java3D.

The functioning of loaders is rather vague.

[18] Consequently, the VRML architecture is simpler than the Java3D one.

6.5.1.3 Discussion

The obvious question we may ask is : " *Are Java3D and VRML97 really in competition?* "

Java3D is not really in competition with VRML. As a matter of fact, the two are largely complementary !
As overviewed in the introduction, VRML is predominantly **a file format for 3D models** on the Web whereas Java3D is predominantly **a 3D graphics runtime system:**

- Specialized editors - like VRCreator 2.0 or Cosmo Worlds - can help to create complex VRML contents. These contents may be then loaded into the Java3D runtime with one of the various loaders available for free on the Web.
  VRML is certainly not the only choice for three-dimensional content, but it is a good one ! It is an ISO standard since 1997 and it was designed with the Web in mind, hence it is fairly compact and browser-friendly.
- On the contrary, Java3D does not define a file or network format of its own. It is designed to scale well and provide support for markets that require **higher levels of performance** that VRML is not able to provide; specifically real-time games, sophisticated mechanical CAD applications, and so on. This is a tool for professionals !
  Java3D is a runtime programming API, first and foremost. Therefore it is not designed to model realistic and complicated three-dimensional world. That is why the loaders were created, in order to use specialized modelling programs.

Therefore, our first choice was to build our objects in VRML, control them and create the application in Java/Java3D. Theoretically, it would be easy since they were designed to work together.

Nevertheless, we have decided to give up Java3D:

- the syntax of Java3D is too restricting.
- the functioning of the loaders is vague
- we should design our own navigation tool.
- the higher levels of performance that Java3D can provide are not required for our Business Visualization purposes.
- Java3D was not adopted by the market as a Business Information Visualization platform.

6.5.2 Our proposal

This technological choice aimed to select the most appropriate technology for the implementation platform of our project. According to us, the implementation platform should allow to develop - or provide - these four components in order to develop a fully functional VR BIV application:

1. A visualization environment
2. A navigation tool
3. A graphical user interface (GUI) for real-time modifications and dynamic queries
4. A High-Level API for dynamic generation of charts and behaviours programming facilities

According to our requirements list for Business Visualization purposes, we have selected a joined development with VRML97 and Java. Using Java in conjunction with VRML97 allows the development of sophisticated, robust, multithreaded applications as well as dynamic, interactive, three-dimensional contents.

The final proposal is:

- for the visualization environment: VRML97
- for the navigation tool: the dashboard provided by the VRML browser
- for the Graphical User Interface: Java (JDK 1.2 - AWT/Swing)
- for the dynamic charts generation and behaviours programming: a High-Level API written in Java (via Internal Script Node of VRML97)

As a matter of fact, several scientists and business visualization softwares believe firmly in VRML

- [Schweber98] submits that VRML should live inside industry standard databases as a fully supported complex data type and be supportable wherever other contemporary data types are supported.
- [EM798a] developed various solutions to develop Business 3D applications. *Portal* is an ActiveX controls API for Business Information Visualization. The developer may specify all of the attributes and characterisitics of a chart and *Portal* produces the corresponding VRML. Since *Portal* is an ActiveX OCX, it is fully programmable and can be

  - embedded into a Visual Basic, Visual C++ or Visual J++ application
  - used in Excel through VBA to graph a range of numbers in a spreadsheet
  - included in a server side Active Server Page (ASP) script to produce graphs dynamically.

- [EM798b] also developed *VRDev*. *VRDev* is a set of Microsoft COM classes that allow developers, who are not VRML experts, to create any VRML content with any COM compliant programming language.
- [Online98] proposes the *Data Visualization Framework* (DVF). DVF is a general approach to map data from ODBC-compliant databases to 3D geometric representations and is geared towards viewing visualizations across the Internet. That is why DVF incorporates the two network-centric technologies we chose: VRML and Java.

6.6 Conclusion

In order to develop the four components that we have defined as most efficiently as possible, we have presented the technological requirements which seem important to us and that the technology chosen for the implementation platform must fulfill. Furthermore, in order to develop our own application, we have checked the requirements compliance of VRML97 and Java3D.

As the first three components already exist, we just have now to design the High-Level API architecture for dynamic generation of charts (VR BIV API). This is the topic of the next chapter.

Chapter 7

Proposal of a Logical Architecture for the VR BIV API

7.1 Introduction

We aim to design a High-Level API for the programming of dynamic generation of charts and of some behaviours. Remind us that a High-Level API for dynamic generation of charts is the fourth component of our definition of a VR BIV application.

We shall propose in this chapter a logical architecture designed during our training at the University of Port Elizabeth, under the supervision of Professor Janet Wesson.

A key feature of this architecture must be its great flexibility. As a matter of fact, this architecture is designed to be extended[1] easily. In this sense, our architecture should be a framework[2].

Typically, the main classes of a framework can be easily reused - or easily extended - to solve problems in different contexts. For instance, classes designed to store business data can be used for various database-centric applications. A framework features a flexible architecture. However, our architecture suffers from some shortcomings that we shall discuss in section 7.3.3 regarding the evolutivity of our API.

[1] in order to add symbolic primitive objects and chart objects

[2] Note that our architecture is *almost* a framework

7.2 Abstract Data Type (ADT) overview

Our API architecture is based on the *Abstract Data Type* theoritical model.

An *Abstract Data Type* (ADT) is a collection of data and related operations. It is defined by describing the logical structure of the information represented by an instance of the type and the operations performed on the information. Moreover, an ADT is about **publicly** how to give access to that data but **privately** how to organize data [Wijngaards98].

The user knows **what** the ADT can do but **not how** it can do it. He only knows which data he has to give to his ADT, which operations he can perform on them and which results the ADT will return to him. Therefore, an ADT provides:

- *Abstraction*: a type's abstract state can be independent of its physical representation
- *Black box*: private data and public operations
- *Encapsulation* with information hiding
- An *interface* for the outside world

Its main advantages are modularity, generality and protection.

Objects are a very good way to implement ADT's since Object Oriented Programming (OOP) implementation of ADT's enhances reusability and improves information hiding.

Implementing an ADT means:

- to decide on internal structure of the type
- to design algorithms for operations
- to design and specify the programming interface

The implementation we are dealing with requires ADT returning *symbolic chart objects* representing data set.
We shall define a High-Level interface showing data values of the representation to the final user and not the physical features (this part is hidden in the implementation). For instance, the user enters his data (company results, and so on) and the ADT builds the representation using a VR Column Chart. If the user modify a mark via the interface, a physical feature of the column will change to reflect the new state.

7.3 The presentation of the logical architecture

7.3.1 The four levels of our API architecture.

In order to design an efficient API architecture, we chose to follow the rules of the ADT paradigm. Among others, we thought that splitting the API architecture into four fundamental levels would be a good way to achieve this efficiency purpose.

1. **First level** *:* the *data* classes
2. **Second level** : the *symbolic primitive objects* classes and *additional components* classes
3. **Third level** : the *symbolic charts* classes
4. **Fourth level** : the t*ranslation* class

A rough diagram of the four levels are illustrated in figure 7.1 (page 124). Note that we draw all our diagrams with the UML[3] language.

7.3.2 The description of the VR BIV API

7.3.2.1 Description of the level

First level: *data* classes
The first level concerns the storage of the business data values that have to be visualized. Each data dimension will be mapped on a primitive dimension or a primitive visual cue.

[3]Unified Modelling Language

The **main** classes of this level are :

- **ListData** to store non-geometric data
- **ListHierarchyData** to store geometric hierarchical data
- **ListNetworkedData** to store geometric networked data
- **ListMatrixData** to store geometric matrix data

Second level : *symbolic primitives* and *additional components* classes
The second level concerns the design of the objects **symbolizing** the elementary data components as well as the basic non-data components of a chart. We distinguish two categories of classes : the *symbolic primitives* classes and the *additional components* classes.

The objects of the s*ymbolic primitives* classes will be instantiated with the objects values of the *data* classes. They constitute the elementary data components used in the representations generation (see section 5.2).
The objects of the *additional components* classes are the basic non-data components making the chart readable, usable and efficient (see section 5.3).

The two main classes of this level are :

- **Primitive** which contains the common properties of the primitive objects. All the specialized *symbolic primitive* classes (e.g. Ribbon.class, Sphere.class) will inherit from it.
- **AdditionalComponent** which is the superclass of all the specialized *additional components* classes (e.g. Background.class, Light.class, Viewpoint.class).

Third level: *symbolic charts* classes
The third level concerns the design of the objects symbolizing the charts.

The objects of the *symbolic charts* classes will logically aggregate the *symbolic primitives* objects and the *additional components* objects to build a complete chart (e.g. : Column Chart, Tree3D, ...)

The main classes of this level are :

- **NonGeometricChart** representing the data of a **ListData** object symbolically
- **HierarchicalChart** representing the data of a **ListHierarchyData** object symbolically
- **NetworkedChart** representing the data of a **ListNetworkedData** object symbolically
- **MatrixChart** representing the data of a **ListMatrixData** object symbolically

Fourth level : *translation* class
The fourth level concerns the conversion of the third level objects in three-dimensional models that can be used within a virtual environment..

The objects of the *translation* class passes on the *symbolic charts* objects to the virtual environment The class used in this level is :

- **Draw**

Note
The level structure provides a great reusability:

- Since only the fourth level is connected to the virtual environment, only this level has to be redesigned if different virtual environments (e.g. VRML, Java3D) are used.
- Since the *symbolic charts* classes are split from the *data* classes, the charts do not have to take care themselves of the data storage. Moreover, in case of a data set is modified, only the *data* object that stores this set has to be updated.
- Since the first level classes are independent from the other levels, the data objects can be used in various uses, such as different types of applications, database integration, and so forth.

7.3.2.2 Implementation of the classes

We are now going to describe how the classes and their relationships have been implemented.

1. First level

   (a) **ListData** consists of a list of **Data** objects.

   (b) **Data** is an array of **Attribute** objects

   (c) **Attribute** is a value (integer, string) which will be mapped on one visual attribute of a primitive object (colour, size, transparency, and so on). All attributes possess a default value which can be modified by the business data entered by the user.

   (d) **ListHierarchyData** consists of a list of **HierarchyData** objects

   (e) **HierarchyData** is a set of all the data relative to a hierarchy structure node.
   A **HierarchyData** object is characterized by two data types:

      i. a **Data** Object

      ii. data determining its situation in the hierarchy: the parent name, the position, the level number, the number of children, and so forth.

   (f) **ListNetworkedData** consists of a list of **NetworkedData** objects

   (g) **NetworkedData** is a set of all the data relative to a network structure node.
   A **NetworkedData** object is characterized by two data types:

      i. a **Data** Object

      ii. data determining the list of related nodes.

   (h) **ListMatrixData** consists of a list of **MatrixData** objects

   (i) **MatrixData** is a set of all the data relative to a matrix structure element.
   A **MatrixData** object is characterized by two data types:

      i. a **Data** Object

      ii. data determining its situation in the matrix : the coordinates in the matrix, a possible content or hyperlink, and so forth.

   Figure 7.2 (page 125) illustrates the first level.

2. Second level

   (a) The **Primitive** class defines all the common attributes of a chart primary component (e.g. colour, transparency, and so on).
   Primitive is the superclass of all the specialized *symbolic primitive objects* classes. **Cube**, **Sphere**, **Ribbon** and all the other specialized *symbolic primitive objects* classes define their own specific attributes (e.g. primitives dimensions like the radius and the height for a cylinder).

(b) The **AdditionalComponent** class defines all the common attributes of a chart *additional components* (e.g. coordinates in the chart). **AdditionalComponent** is the superclass of all the specialized *additional components* classes. **Background**, **Label**, **Grid** and all the other specialized *additional components* classes define their own specific attribute. For instance, the attributes of **Background** are the sky colour, the ground colour, the sky angle, the ground angle, and so on.

Figure 7.3 (page 126) illustrates the second level.

3. Third level

(a) The **NonGeometricChart** class defines the common attributes of non-geometric charts.
A **NonGeometricChart** object is characterized by two data types:

i. a **ListData** object.

ii. the particular chart attributes (e.g. the coordinates in the virtual environment)

NonGeometricChart is the superclass of all the specialized non-geometric charts classes (e.g. **ColumnChart** class, **ScatterChart** class, **RibbonChart** class, and so forth).

(b) The **GeometricChart** class defines the common attributes of geometric charts (e.g. the coordinates in the virtual environment). **GeometricChart** is the superclass of three classes symbolizing three charts subtypes: the **HierarchicalChart** class, the **NetworkedChart** class and the **MatrixChart** class.

The **HierarchicalChart** class, the **NetworkedChart** class and the **MatrixChart** class were designed following the same pattern. However, in an exhautiveness concern, we are providing the description of each class anyway.

HierarchicalChart
The **HierarchicalChart** class defines the common attributes of hierarchical charts.
A **HierarchicalChart** object is characterized by two data types:

i. a **ListHierarchyData** object

ii. the particular attributes of the chart (e.g. the nodes list)

HierarchicalChart is the superclass of all the specialized hierarchical charts classes (e.g. **Tree3D** class, **InfoCube** class, and so on).

NetworkedChart
NetworkedChart defines the common attributes of networked charts.
A **NetworkedChart** object is characterized by two data types:

i. a **ListNetworkedData** object
ii. the particular attributes of the chart (e.g. the nodes list)

NetworkedChart is the superclass of the **InfoBall** class.

Note that designing a superclass for only one subclass is a design choice that speaks for itself. It allows to extend the architecture easily by additional subclasses.

MatrixChart
The **MatrixChart** class defines the common attributes of matrix charts .
A **MatrixChart** object is characterized by two data types:

i. a **ListMatrixData** object
ii. the particular attributes of the chart (e.g. the elements list)

MatrixChart is the superclass of the **DataCube** class.

Figure 7.4 (page 127) illustrates the third level.

4. Fourth level
Since the modelling way depends on the chart class, the **Draw** class knows how to draw each charts.
Draw is a class which can model any chart or primitive object in the selected virtual environment. The **Draw** object is performing three tasks:

(a) matching symbolic primitive objects to the virtual environment primitives
(b) matching additional components to the virtual environment primitives
(c) making fit the primitives and the additional components together

Since the fourth level is connected to the virtual environment, methods of the **Draw** class must be adapted to this environment. This means that chosing a different virtual environment implies the methods rewriting .
Figure 7.5 (page 128) illustrates the fourth level.

Figure 7.6 (page 129) illustrates the complete API architecture. As this diagram is rather complex, a A3 version is provided in Appendix D.

7.3.3 Evolutivity of the High-Level API

We have just seen that the **Draw** object is performing three tasks:

1. matching symbolic primitive objects to the virtual environment primitives
2. matching additional components to the virtual environment primitives
3. making the primitives and the additional components fit together

The success of those tasks is possible **only if the virtual environment provides all the required primitives**. But a virtual environment provides often a limited set of preexistent primitives[4]. This could be a problem for the API design and its evolutivity. That is why the API architecture must be flexible, in order to be able to develop new primitive objects drawable in the virtual environment, new symbolic primitive objects and new symbolic charts.

We may then sum up the design and extension process of the API in four stages :

- the extension of the virtual environment primitives, **if necessary**
- the design of their symbolic object counterparts, i.e. the design of the *symbolic primitive objects* and *additional components* classes (see second level in section 7.3.2), **by adding classes**
- the design of the *symbolic charts* classes (see third level in section 7.3.2), **by adding classes**
- the design - or extension - of the *translation* class (see fourth level in section 7.3.2), **by *modifying* the *Draw* class**

To prove that this architecture was working properly, we have implemented and tested the following classes at UPE:

First Level:

- Attribute.java
- Data.java
- Listdata.java

[4]For instance, VRML proposes only four primitives: the box, the cone, the cylinder and the sphere.

- HierarchyData.java
- ListHierarchyData.java

Second level:

- Primitive.java
- Cube.java
- Sphere.java

Third level:

- HierarchicalChart
- Tree3D
- NonGeometricChart.java
- ColumnChart.java

Fourth level:

- Draw.java

The code of the Java classes is available in Appendix D and is joined with an A3 version diagram of the complete API architecture.

Criticism of the evolutivity of our architecture

The last step of the extension process, i.e. the design - or **extension** - of **the** ***translation*** **class**, is one weak point. We mentioned in the introduction that our architecture is not completely a framework. Typically, the main classes of a framework can be easily extended just by adding classes. However, we have to modify the *translation* class[5] because only this class knows how to draw each charts. This is a complication !

A real framework would not modify this *Draw* class. Each adding *symbolic charts* class should know how to draw itself and "explain" it to the *Draw* class. In this case, the *Draw* class would be a "factory" being able to draw any charts if it receives the appropriate instructions.

[5]the fourth level

7.4 Conclusion

We have right now presented the four levels of our API architecture. Although this architecture suffers from some shortcomings, it was designed with VR BIV purposes in mind.

The proposal of this architecture was our last requirement in the determination of the general design process requirements for the development of a VR BIV Application. To check the validity of the requirements of our design process, we have programmed a little VR BIV application. The application presentation is the topic of the final chapter.

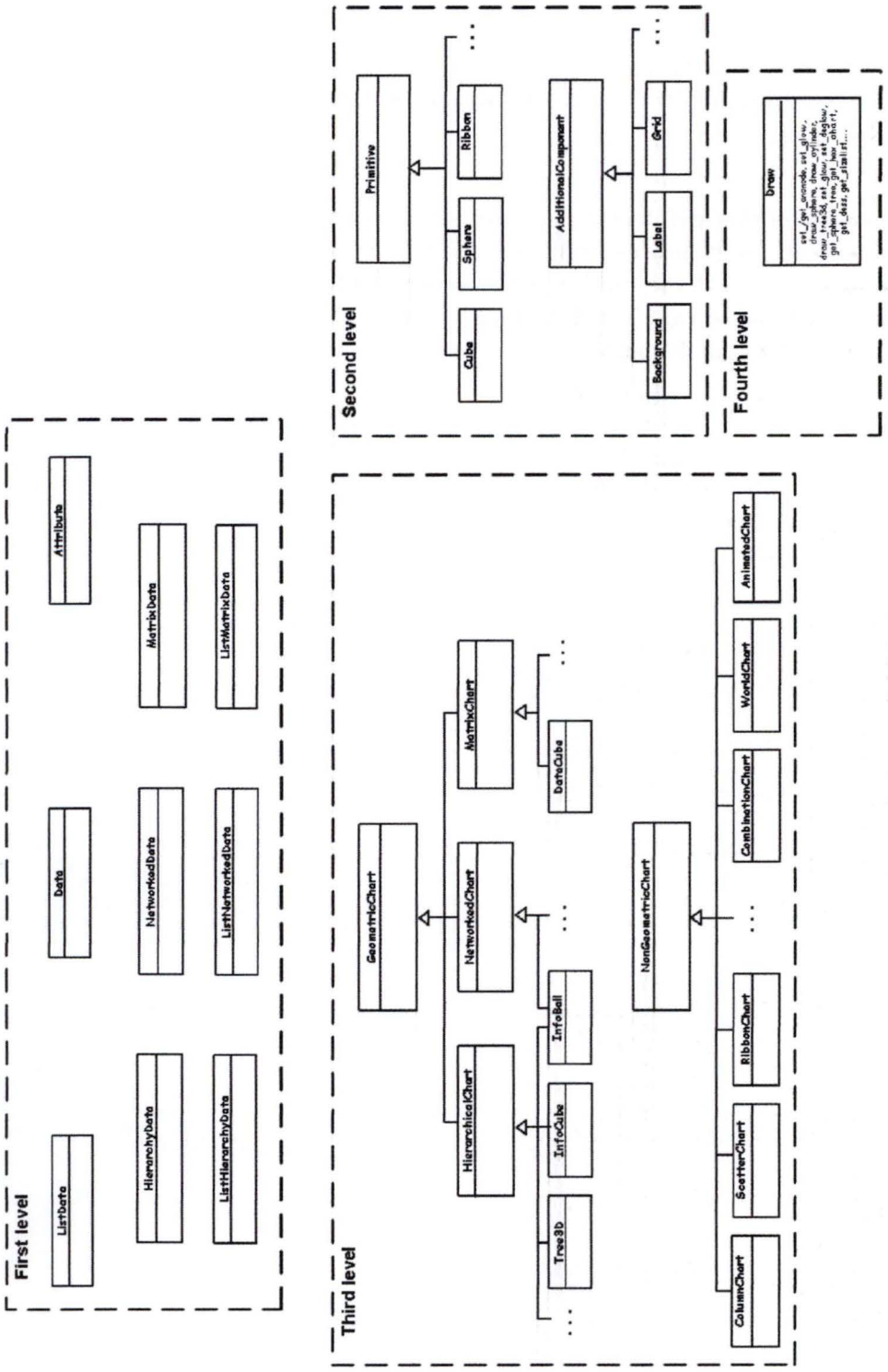


Figure 7.1: An rough overview of the four levels of the architecture

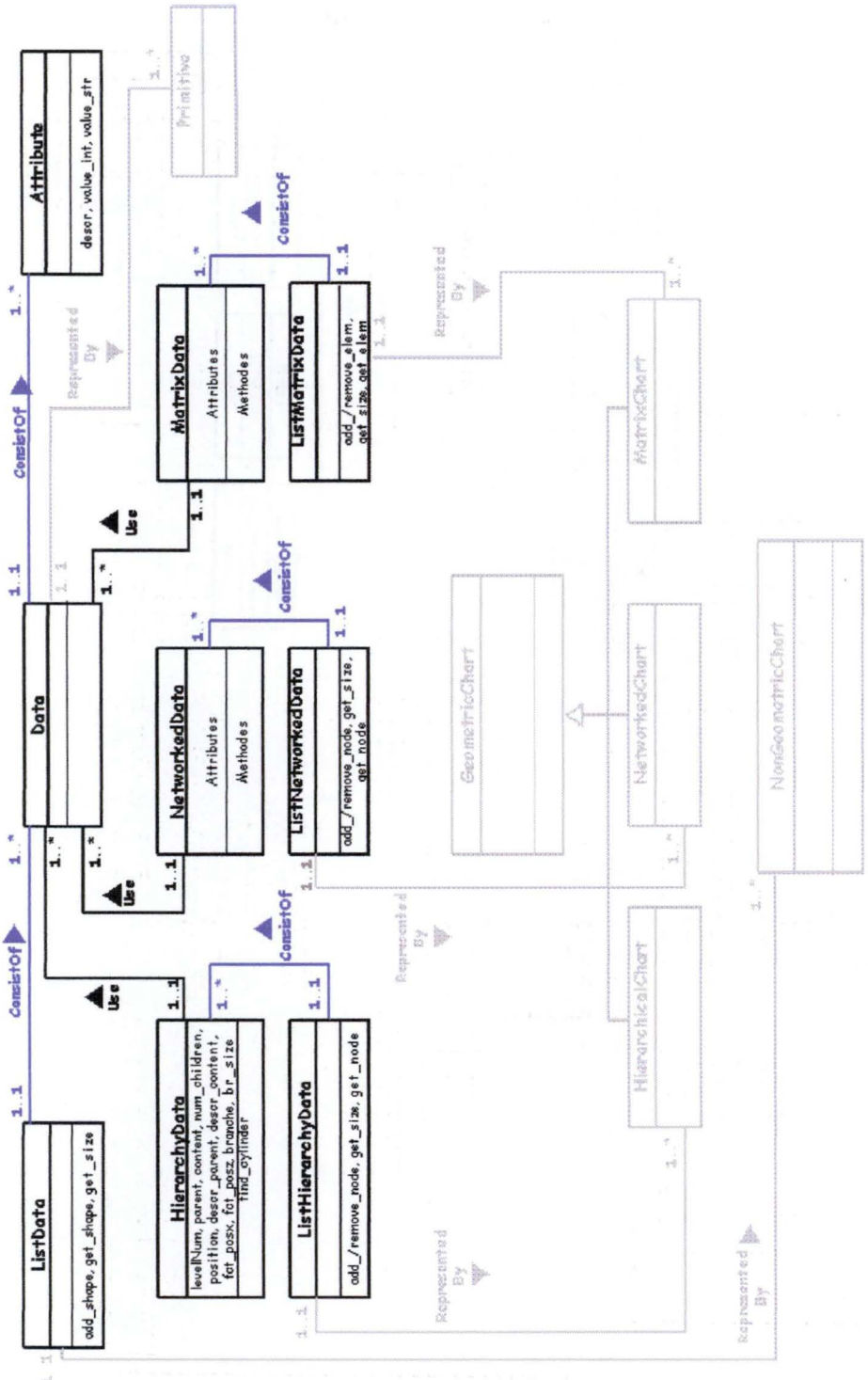


Figure 7.2: The first level of the API architecture

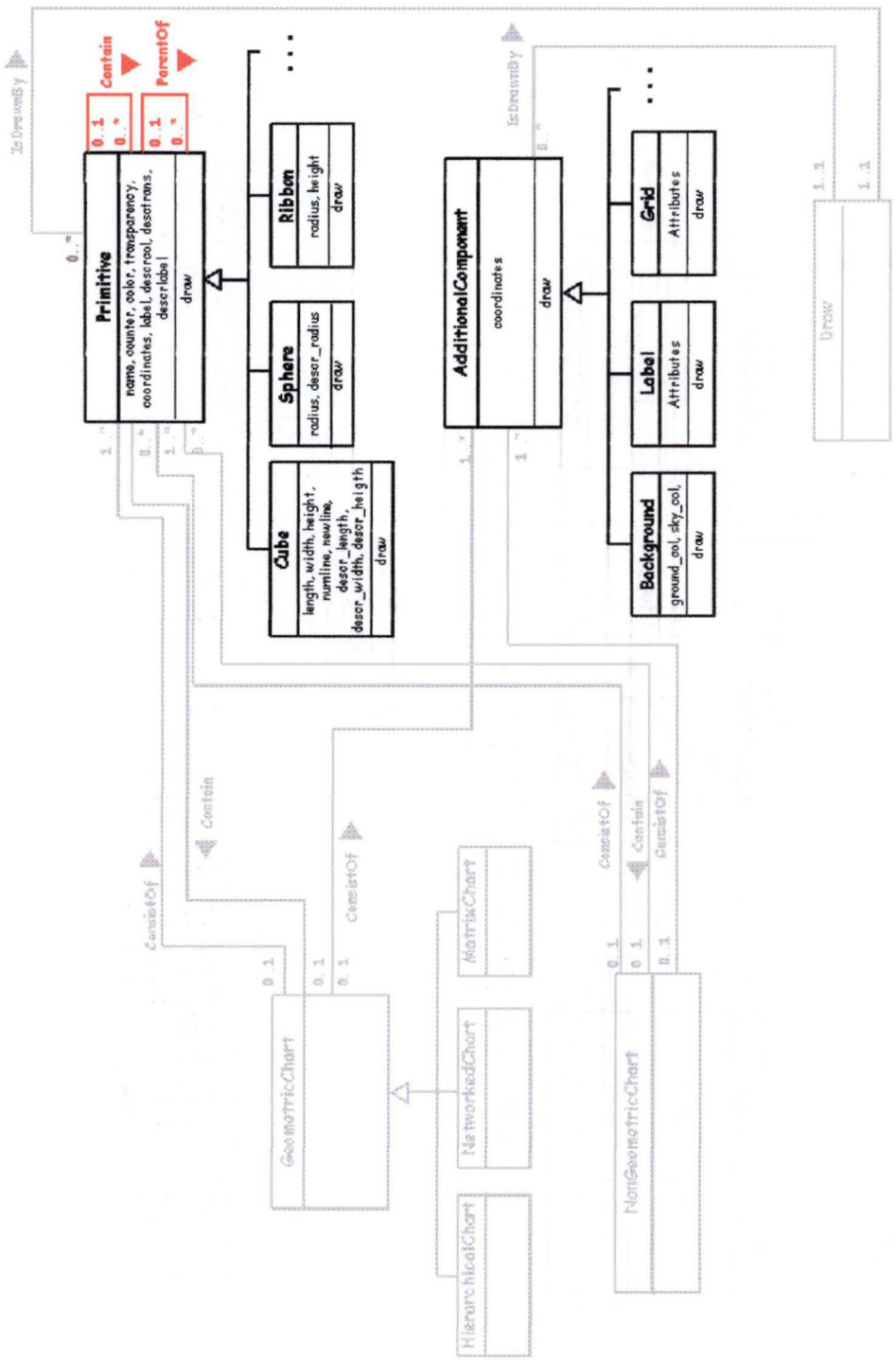


Figure 7.3: The second level of the API architecture

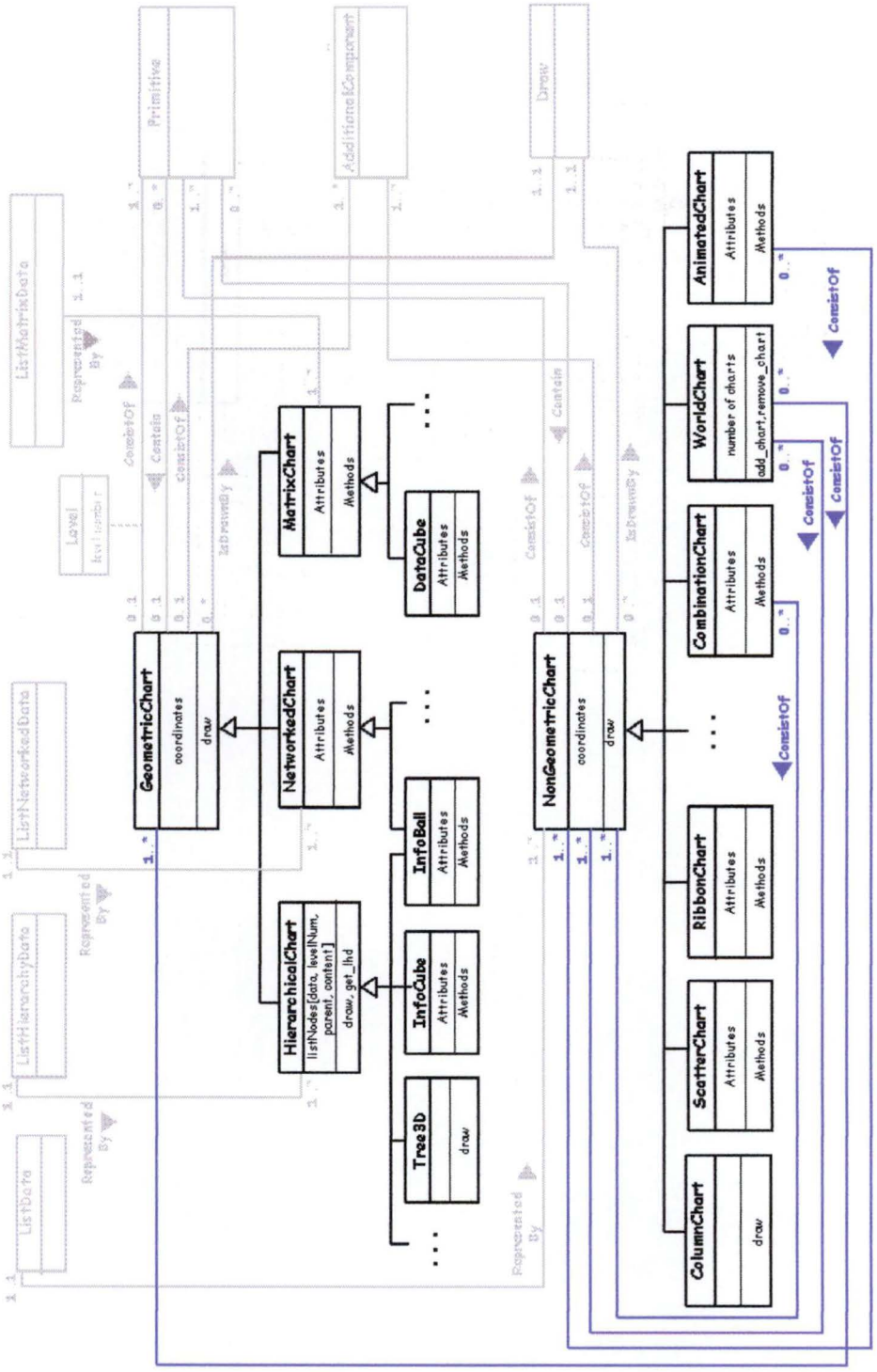


Figure 7.4: The third level of the API architecture

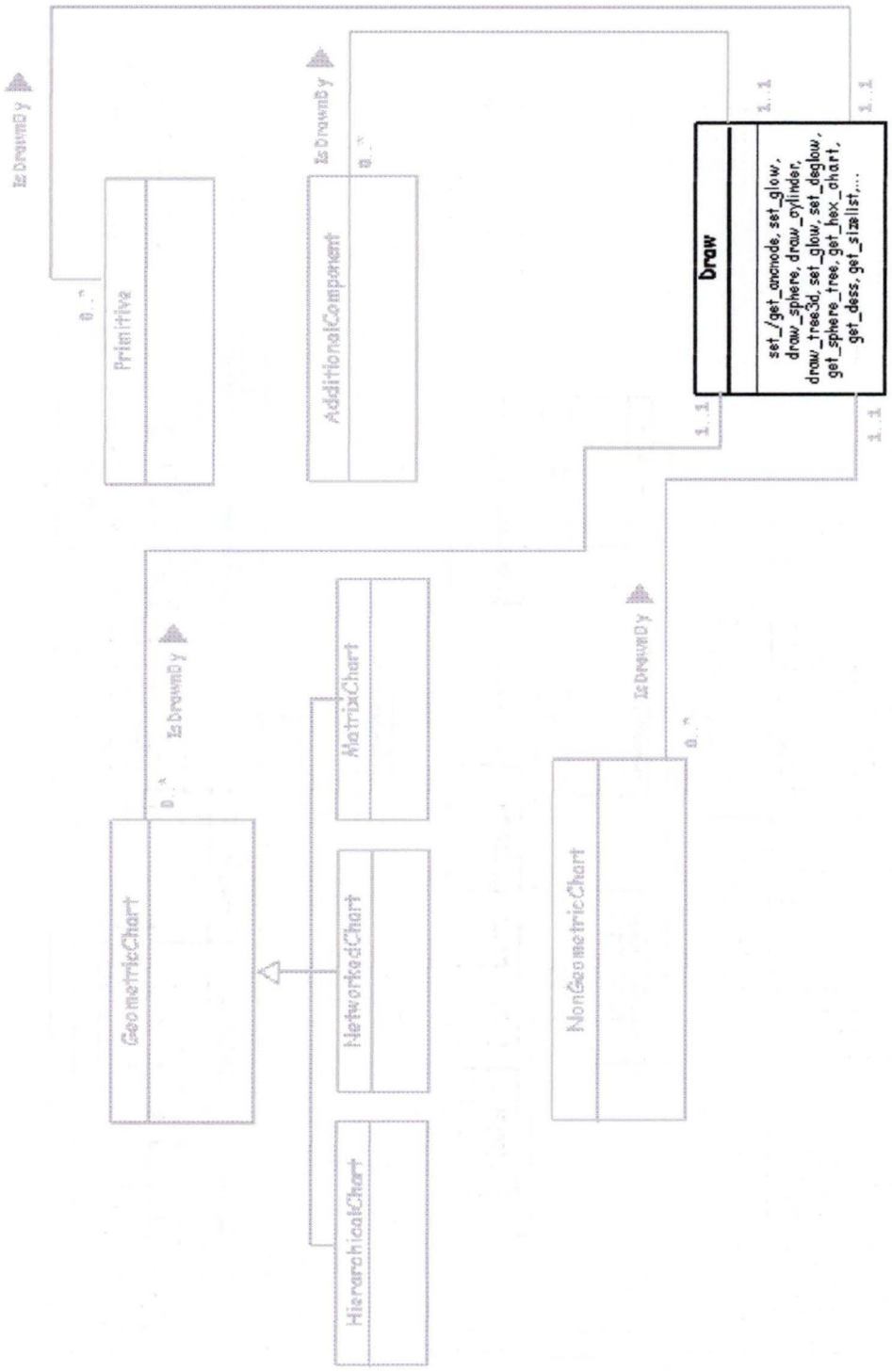


Figure 7.5: The fourth level of the API architecture

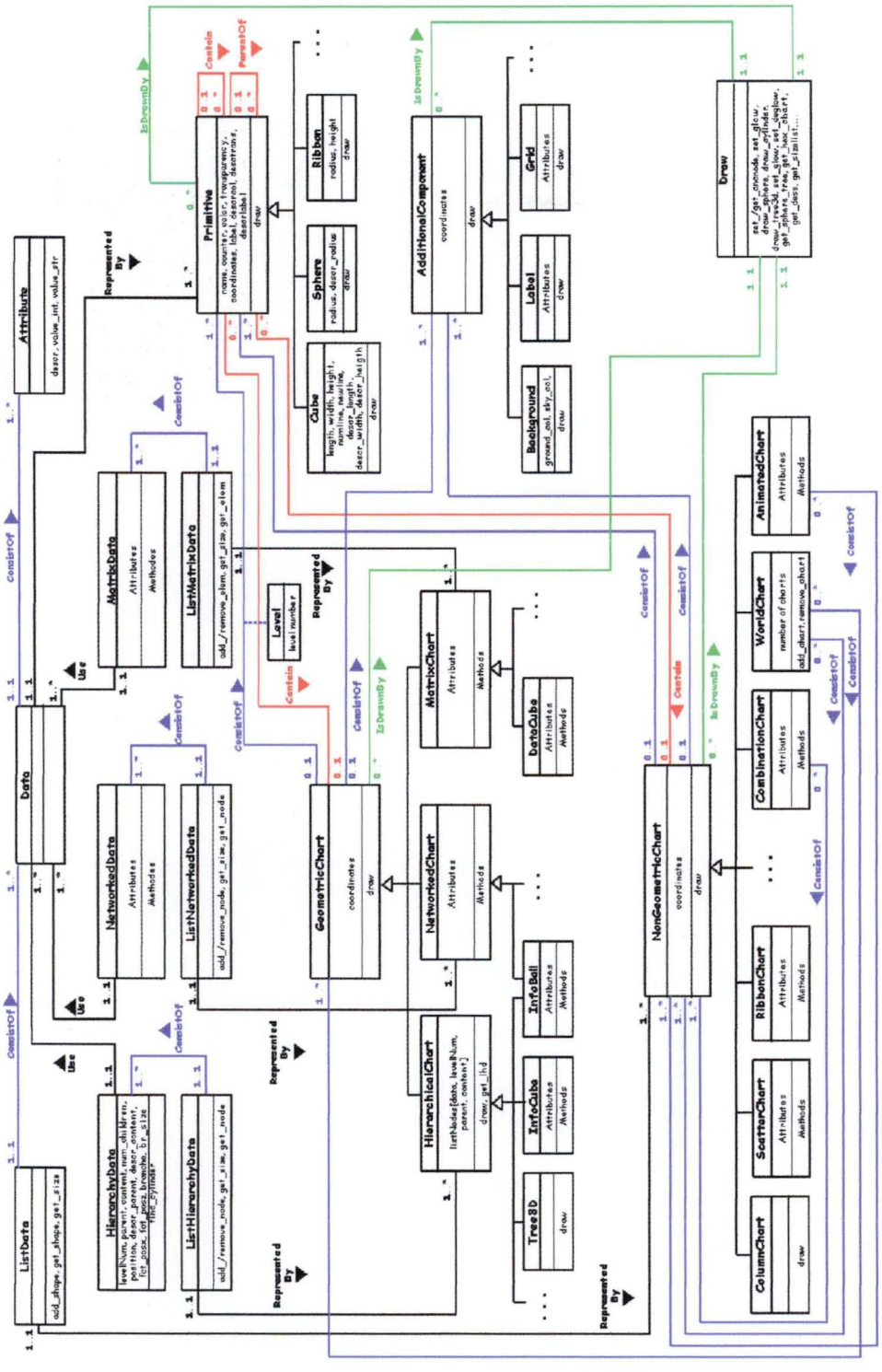


Figure 7.6: The complete API architecture

Chapter 8

Application Example

8.1 Introduction

To check the validity of the requirements of our design process, we have programmed a little VR BIV application based on the classes we had implemented at the University of Port Elizabeth.

8.2 Aim of the application

The aim of this application is to visualize hierarchical data. The user has to enter its data via a Graphical User Interface (see figure 8.2, page 134). When the user clicks on the "*Build the tree*" button, the dynamic building of the chart is generated by the application. A Tree3D is then displayed in the visualization window (see figure 8.3, page 134).

Working live: Population of Namur			
Male		Female	
< 25	25-60	<25	25-60
10%	89%	8%	77%

Table 8.1: The data set visualized by our VR BIV application

The data set visualized by our VR BIV application is shown in table 8.1.

By using the navigation tool, the user may study the hierarchical data comfortably. If the generated Tree3D is too voluminous, a list box allows to highlight[1]

[1] thanks to a glowing effect

a specific node to locate it within the hierarchy.

We admit that the "blue" GUI is not particularly well designed but we suffer from a lack of time at the end of our training, since the API development has taken a lot of time.

8.3 Technological choice

- for the visualization environment: VRML97
- for the navigation tool: the dashboard provided by the Cosmo Player
- for the Graphical User Interface: Java (JDK 1..1.5 - AWT)
- for the dynamic generation of chart and behaviours programming: a High-Level API written in Java (via External Authoring Interface)

Figure 8.1 shows the four components clearly.

NOTE:
The application is running into an applet (interface.java). Therefore, we used the JDK 1.1.5 since it is the release of the virtual machine of the current HTML browsers.

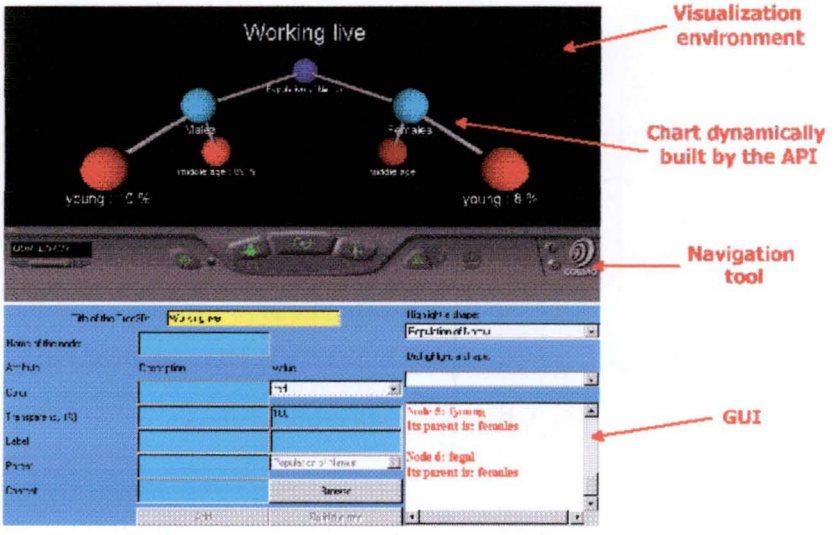


Figure 8.1: The four components of a VR BIV application

We used the EAI instead of Internal Script Node since we had no command of ISN at this moment.

8.4 Architecture

The classes of the following levels were used to develop this application: **First Level:**

- Attribute.java
- Data.java
- HierarchyData.java
- ListHierarchyData.java

Second level:

- Primitive.java
- Sphere.java

Third level:

- HierarchicalChart
- Tree3D

Fourth level:

- Draw.java

***NOTE*:**
To make our task easier, the edges (represented by the *Cylinder* primitives of VRML97) were precalculated in a VRML file (root.wrl). Their attributes are controlled by the *Draw* class. They are made visible only to show parent-child relationships.

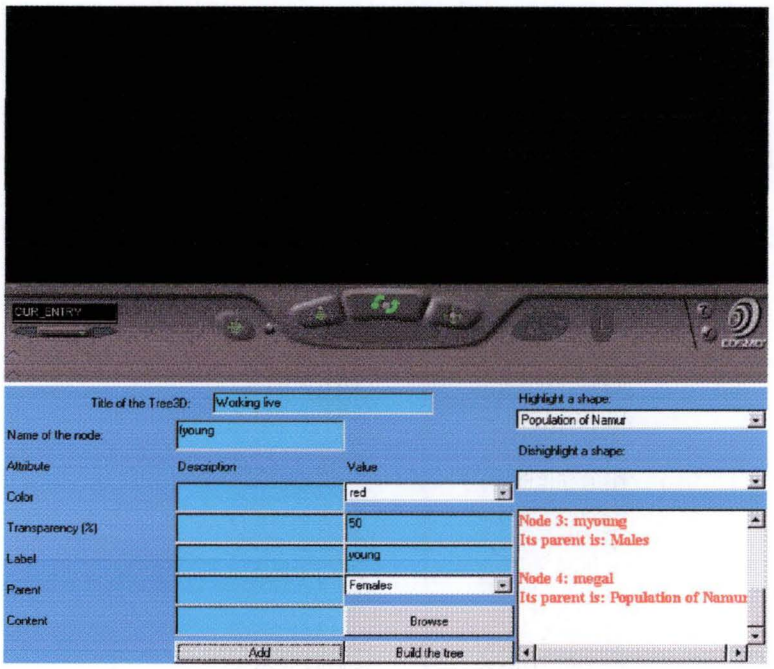


Figure 8.2: The user is entering data

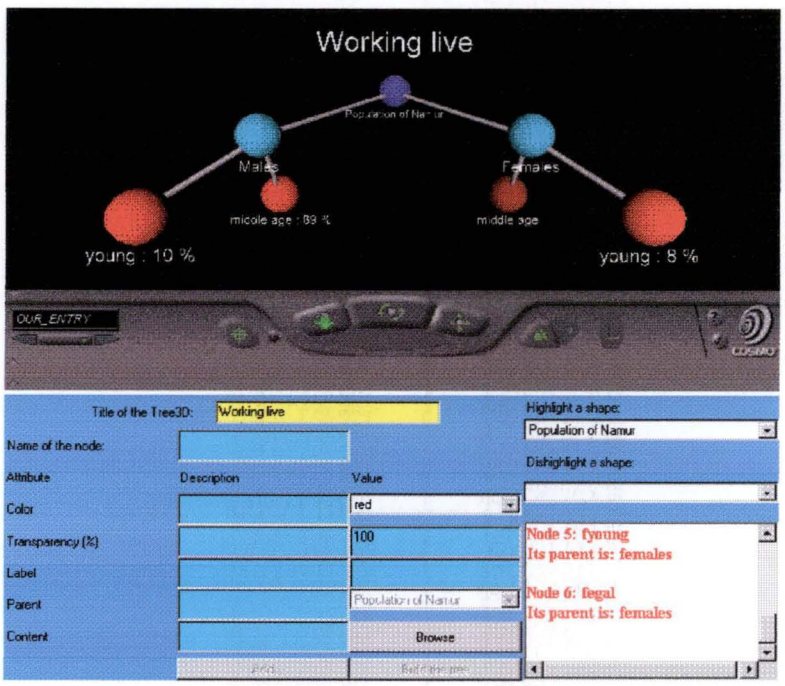


Figure 8.3: The display of the chart.

Part III

Conclusion

Conclusion

The scope of our thesis was the study of the Virtual Reality (VR) technology used in the visualization of Business Information (BIV).

Business Information Visualization is a process of creating appropriate computer-generated visual representations of large amount of non-geometric managerial data for human problem-solving and decision-making support [Zhang99b].

As Business Information Visualization is a relatively new field in which few theoretical and practical results have been obtained, our first aim was to analyse how far VR could be an efficient platform to deal with the challenges raised by Business Information Visualization.

Our second aim was the determination of the general design process requirements for the development of a VR BIV application.

This thesis was subdivided into two parts, each one covering an aim. The first one focused on the analysis of the Virtual Reality usefulness in Business Information Visualization field. The second one focused on our researches on the design process requirements for the development of a VR BIV application.

Part 1
In the first chapter, we have presented the different fields using the Virtual Reality technology briefly. Next, we dealt with the set of problems coming with decision-making as well as the various tools designed to solve them.
In chapter 3, we have introduced the semantic properties of business data and we portayed the most common VR charts. Finally, we concluded by a discussion about assets and shortcomings of using the Virtual Reality for the Business Information Visualization field.

In conclusion, we showned in the first part that the Virtual Reality technology has a future in complex problem-solving and decision-making. The Visual User Interface, the best cost-effectiveness of the screen space, the various visual cues and the real-time visualization are the keys of the Virtual Reality power which convey information very efficiently as well as provide interactive capabilities to exploit the maximum of business data.

However, analyses using VR are new to business environments and many researches have to be led in order to settle human visual perception problems and the interface usability.

Part 2
In the fifth chapter, we have reviewed the list of the most common objects composing the VR charts. With this in view, description of geometric primitives and additional components constituting VR Charts were overviewed. Afterwards we have discussed some technological requirements that the implementation platform must fulfill. Finally we have proposed a possible architecture designed for the creation of VR BIV applications

In conlusion, by developing our own example of a VR BIV application, we have proved the validity of the three general requirements of the design process that we defined. These three general requirements are:

- the list of the most common objects composing the VR charts
- some technological requirements that the implementation platform must fulfill
- a possible architecture designed for the creation of VR BIV applications

Finally, this thesis will be entirely completed only by few words about our last university year.

This last and fifth year distinguished itself by two enriching experiences. The first experience is the training at the University of Port Elizabeth and the second is the making of this thesis.

The work achieved at UPE, under the supervision of Professor Janet Wesson and Senior Lecturer Leon Nicholls, was really beneficial. Furthermore, the contact with people whose the culture is so different as well as the beauties and the mysteries of South Africa were great experiences.

This thesis learned us how to write a serious report. The research, the ideas development, the writing are big issues while making a thesis. We tried to make it in a professional way. We hope that we took up the challenge with success !

Bibliography

[About97] ABOUT, "Financial Visualization in VRML", *http://web3d.about.com/compute/web3d/library/weekly/aa121597.htm?iam=mt&terms=%2Bbusiness+%2Bvisualization*, 1997

[AdvancedVisual98] ADVANCED VISUAL SYSTEM INC., "Openviz : Business Visualization Technology", *http://www.avs.com/openviz*, 1998

[AlterVue98] ALTERVUE SYSTEMS INC., "VRCharts : The First Practical Application of Interactive 3-D Data Visualization Software For Widespread Business Use", *http://www.vrcharts.com/demo*, 1998

[AWT00] AGENCE WALLONE DES TÉLÉCOMMUNICATIONS, "Le Business to Consumer (B to C). Critčre de définition 1: la nature des produits", *http://www.awt.be*, 2000

[Bailey99] DUANE A. BAILEY, "Java Strucures. Chapter1 : The Object-Oriented Methods", *McGRAW-HILL INTERNATIONAL EDITIONS, Computer Science Series*, 1999

[Bell95] G. BELL, A. PARISI & M. PESCE, " The Virtual Reality Modeling Language ", *http://www.mwc.edu/~pclark/intro.html*, May 26 1995

[Bell97] JAMES BELL, "The Web Enters The Third Dimension", *http://www.winmag.com/library/1997/0901/windo120.htm*, 1997

[Bell-Foggler95] JOHN T. BELL AND H. SCOTT FOGLER, "The Investigation and Application of Virtual Reality as an Educational Tool", *Proceedings of the American Society for Engineering Education 1995 Annual Conference, Session number 2513, Anaheim, CA*, 1995

[Bodart-Magnier99] Th. Bodart, M.L. Magnier, "A Multimedia Support to the Learning of Interaction Objects", master thesis, Institut d'Informatique, FUNDP, Namur, 1999

[Bouvier99] D. J. BOUVIER, " Getting Started With Java 3D API. Chapter 1 ", *http://java.sun.com/products/java-media/3D/collateral*, October 08 1999

[Bryson99] T. BRYSON, " Exploring the Java 3D API ", *http://www.performancecomputing.com/features/9904f1.shtml*, April 1999

[Carrey-Bell97] RICK CARREY and GAVIN BELL, *The Annotated VRML Reference Manual*, ADDISON-WESLEY DEVELOPPERS PRESS, An imprint of Addison Wesley Longman, Inc, Reading, Massasuchets * Arlow, England * Menlo Park, California Berkeley, California * Don Mills, Ontario * Sydney * Bonn * Amsterdam * Tokyo * Mexico city, 1997

[Chuah-Eick98] MEI C. CHUAH AND STEPHEN G. EICK, "Information Rich Glyphs for Software Management Data", *IEEE Computer Graphics and Applications*, vol.18, No. 4, 1998, pp 24-29

[CompAss99] COMPUTER ASSOCIATES INTERNATIONAL, Inc., "PLATINUM Forest & Trees Technical Overview", *http://www.answer.com/products/busintl/to_ft.htm*, 1999

[Day98] B. DAY, " 3D Graphics programming in Java, Part 1 : Java 3D ", *http://www.javaworld.com/javaworld/jw-12-1998/jw-12-media.html*, December 1998

[Day99] B. DAY, " 3D Graphics programming in Java, Part 2 : Advanced Java 3D ", *http://www.javaworld.com/javaworld/jw-01-1999/jw-01-media.html*, January 1999

[Deering99] M. F. DEERING & H. A. SOWIZRAL, " Frequently asked Questions Java 3D API ", *http://java.sun.com/products/java-media/3D/forDevelopers/java3dfaq.html*, April 06 1999

[Dimension00] DIMENSION 5, "Data Visualization", *http://www.dimensions5.sk/technology.htm*, 2000

[EcoLogic00] ECOLOGIC AND BEAKERWARE, "The Central Hardwoods Virtual Forest", *http://www.snarkware.org/VirtualForest.html*, 2000

[EM798a] EM7 ELECTROHOUSE,"Portal : Guide to the API for VRML Charts and Graphs", *http://www.em7.com*, 1998

[EM798b] EM7 ELECTROHOUSE,"VRDev : Guide to the API for VRML Development", *http://www.em7.com*,1998

[EM700] EM7 ELECTROHOUSE,"Enterprise and Developer Solutions, Business3D Software", *http://www.em7.com*, 2000

[Finlayron99] R. A. FINLAYRON, " The VRML EAI FAQ : VRML 2.0 External Authoring Interface (EAI) FAQ ", *http://www.frontiernet.net/~imaging/eaifaq.html*, June 1 1999

[Garg-Tamassia96] A. GARG AND R. TAMASSIA, "Giotto 3d : A System for Visualizing Hierarchical Structures in 3d", *S. North editor, Proceedings of Graph Drawing '96,* Lecture Notes in Computer Science, to appear, Springer - Verlag, *http://www.cs.brown.edu/cgc/papers/gt-gsvhs-97.ps.gz* , 1996

[Gerding97] D. J. GERDING, " 20 Questions on VRML ", *http://www.builder.com/Authoring/Vrml/?tag=pt.rn.reg.bl*, July 07 1997

[Guru96] RENT-A-GURU, "3D Stat", *http://www.netstore.de/Supply/3Dstats/*, 1996

[Hamilton00] SHARON HAMILTON AND KRISTIN R. FAYER, "Visual Discovery & Reporting : A New Era in Business Decision Making", *http://www.visualinsights.com/pressroom/ whitepapers_visualqa.asp?current=pressroom* , 2000

[Harris99] ROBERT L. HARRIS, *Information Graphics : A Comprehensive Illustrated Reference*, Oxford University Press, New York, Oxford, 1999

[Huai-Hsin98] ED HUAI-HSIN CHI, JOHN RIEDL, PHILLIP BARRY AND JOSEPH KONSTAN, "Principles for information Visualization Spreadsheets", *IEEE Computer Graphics and Applications*, vol.18, No. 4, 1998, pp 30-38

[Jacob96] K. JACOB, " Why Java and VRML ? ", *http://www.javaworld.com/javaworld/jw-03javavrml.html*, March 1996

[Lycos00a] LYCOS INC., "VRML - Tech Glossary", *http://webopedia.lycos.com/TERM/V/VRML.html*, 2000

[Lycos00b] LYCOS INC., "Virtual Reality - Tech Glossary", *http://webopedia.lycos.com/TERM/v/virtual_reality.html*, 2000

[Macpherson98] COLIN MACPHERSON AND MIKE KEPPEL, "Virtual Reality: What is the state of play in education ?", *Australian Journal of Educational Technology* 14(1), 1998 , pp 60-74

[Manovich92] LEV MANOVICH, "Assembling Reality: Myths of Computer Graphics", *http://www-apparitions.ucsd.edu/~manovich/text/assembling.html*, 1992

[Marin97] C. MARIN, " Proposal for a VRML 2.0 Informative Annex : External Authoring Interface Reference ", *http://www.vrml.org/WorkingGroup/vrml-eai/ExternalInterface.html*, November 21 1997

[Microsoft00] MICROSOFT RESEARCH, "The Task Gallery", *http://research.microsft.com/ui/TaskGallery*, 2000

[Mirel99] BARBARA MIREL, "Usability Test Results for Information Visualization : Determinants of Usefulness for Complex Business Problems", *Human-Computer Interaction - INTERACT '99*, Edited by M.A. Sasse and C. Johnsonn IOS Press IFIP TC 13, 1999

[Nielsen98] Jakob Nielsen, "What is Usability ?", http://www.zdnet.com/devhead/stories/articles, September 14, 1998

[Noirhomme00a] MONIQUE NOIRHOMME-FRAITURE, "Multimedia Support for Complex Multidimensional Data Mining", *Congrčs MDM (KDD 2000 Workshop)*, 2000

[Noirhomme00b] M. NOIRHOMME, A. NAHIMANA AND B. DE GREIFT, "Graphic Library: Detailed Functional Analysis for Temporal Star, Simple Star and Urchin ", *ISO-3D Project,* 2000

[OLAP95] THE OLAP COUNCIL, "OLAP and OLAP Server Definitions", *http://altaplana.com/olap/glossary.html*, January 1995

[OLAP97] THE OLAP COUNCIL, "OLAP Council White Paper", *http://www.olapcouncil.org/research/whtpaply.htm*, 1997

[Online98] ONLINE ENVIRONS INC., "Data Visualization Framework", *http://dvf.environs.com*, 1998

[Pilot99] PILOT SOFTWARE INC., "OLAP White Paper", *http://www.pilotw.com/olap/olap.htm*, 1999

[Polevoi98] ROBERT POLEVOI, "3D Animation Workshop", http://webreference.com/3d, 1997-1998

[Rekimoto93] J. REKIMOTO AND M. GREEN, "The Information Cube : Using Transparency in 3d Information Visualization", *Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS '93)*, *http://www.csl.sony.co.jp/person/rekimoto/cube.html*, 1993, pp 125-132

[Ricard99] E. RICARD (MS&I), "State of the Art Report", *ISO-3D Project, Esprit Project (of European Commission)*, 1999

[Roskothen99] MARCUS ROSKOTHEN, *"Projects", http://www.vruniverse.com/projects.html#infoball1*, 1999

[Schneider96] BERNHARD SCHNEIDER, Digital Terrain Modelling Group, "3D Landscape Modelling", *http://www.islandnet.com/lombard*, 1996

[Schobbens98] PIERRE-YVES SCHOBBENS, 'Techniques d'Intelligence Artificielle", *INFO2209,* Chapter 2, section 4: Systčmes f Objets Structurés, Facultés Universitaires Notre-Dame de la Paix, 1998-1999

[Schweber98] ERICK VON SCHWEBER, "Escape from VRML Island", *http://www.infomaniacs.com/SQL3D/SQL3D-Escape-From-VRML-Island.htm*, 22 July 1998

[Siggraph97] SIGGRAPH 97, "Panel on Facial Animation : Past, Present and Future", *http://mambo.ucsc.edu/psl/sig97/siggraph97-panel*, 1997

[Sun99a] SUN MICROSYSTEMS, " The Evolution of 3D Graphics ", *http://java.sun.com/products/java-media/3D/collateral/presentation/*, October 08 1999

[Sun99b] SUN MICROSYSTEMS, " Java 3D API 1.1 Performance Guide ", *http://java.sun.com/products/java-media/3D/collateral/presentation/*, August 19 1999

[Sun99c] SUN MICROSYSTEMS, " Java 3D API, White Paper ", *http://java.sun.com/marketing/collateral/3d_api.html*, April 22 1999

[Sun99d] SUN MICROSYSTEMS, " Java 3D API Specification. Chapter 1 : Introduction to Java 3D ", *http://java.sun.com./products/java-media/3D/forDevelopers/j3dguide/*, 1999

[Sundset97] T. SUNDSET, " 3D computer graphics : Getting the hang of VRML ", *http://www.javaworld.com/javaworld/jw-08-1997/jw-08-howto.html*, August 1997

[ThreeDGraphics00] THREE D GRAPHICS, "Amigo 2000 : Reviewing Guide", *http://www.amigo2000.com*, 2000

[UnivBelfast99] THE QUEEN'S UNIVERSITY OF BELFAST, "Data Mining Notes Data Mining", *http;//www.pcc.qub.ac.uk/tec/courses/datamining/stu_notes/dm_book_2.html*, 1999

[UnivMississipi97] MISSISSIPPI STATE UNIVERSITY, "The List ADT", *http://www.cs.msstate.edu/~cs1314/fall97/slides/linkedlist/index.htm*, 1997

[Verna97] D. VERNA, " Une Introduction f VRML ", *http://www.infres.enst.fr/~verna/enseignement/vrml-fr.html*, Mars 7 1997

[VRMLConsortium97] THE VRML CONSORTIUM INCORPORATED, " The Virtual Reality Modeling Language. International Standard ISO/IEC 14772-1:1997 ", *http://www.vrml.org/Specifications/VRML97*, 1997

[WakeForest00] VIRTUAL ENDOSCOPY CENTER, Wake Forest University School of Medicine, "Virtual Gallery", *http://www.vec.bgsm.edu/gallery/Virtual Endoscopy Center Gallery.htm*, 2000

[Wesson99] JANET WESSON & LESTER COWLEY,"Selecting Interaction Objects: A Pattern Approach", University of Port Elizabeth, Port Elizabeth - South Africa,1999

[Wijngaards98] N. WIJNGAARDS, "Course CDSC 233/L02. Chapter 12 : Abstract Data Types", *http://www.cpsc.ucalgary.ca/~nick/courses/233*, 1998

[Wright98] WILLIAM WRIGHT, "Business Visualization Adds Value", *IEEE Computer Graphics and Applications*, vol.18, No. 4, 1998, p 39

[Zhang99a] DR. PING ZHANG, "Effective Decision Making With Effective Human Information Interaction: A Cognitive Perspective", *Proceeding of the Fourth Asia-Pacific Decision Science Institute Conference,* 1999

[Zhang99b] DR. PING ZHANG, "Business Information Visualization: Guidance for Research and Practice", To be published by *Encyclopedia of Microcomputers & Encyclopedia of Library and Information Science*, 1999

[Zhang-Whinston95] DR. PING ZHANG and ANDREW B. WHINSTON, "Business Information Visualization for Decision-Making Support – A Research strategy", *Proceeding of the First Americas Conference on Information System*, 1995

Part IV

Appendices

Appendix A

Analysis of the Architectures of VRML97 and Java3D

A.1 Introduction

The aim of this technical analysis of the VRML97 and Java3D architectures is to support us in the technological choice made in chapter 6. The analysis result has been foremost significant for the *simplicity* requirement (see section 6.3.2 and 6.4.2) but it also gave us a more detailed knowledge of the capabilities of both technologies.

This analysis was made at the University of Port Elizabeth.

A.2 VRML97 architecture

A.2.1 File structure

A VRML file consists of the following major functional components: the *header*, the *scene graph*, the *prototypes*, and the *event routing*. The contents of this file are processed for presentation and interaction by a mechanism known as a browser (see figure A.1).

[Carrey-Bell97] have described the structure of a VRML file. Each VRML file

- implicitly establishes a world co-ordinate space for all objects defined in the file, as well as all objects included by the file
- explicitly defines and composes a set of 3D and multimedia objects

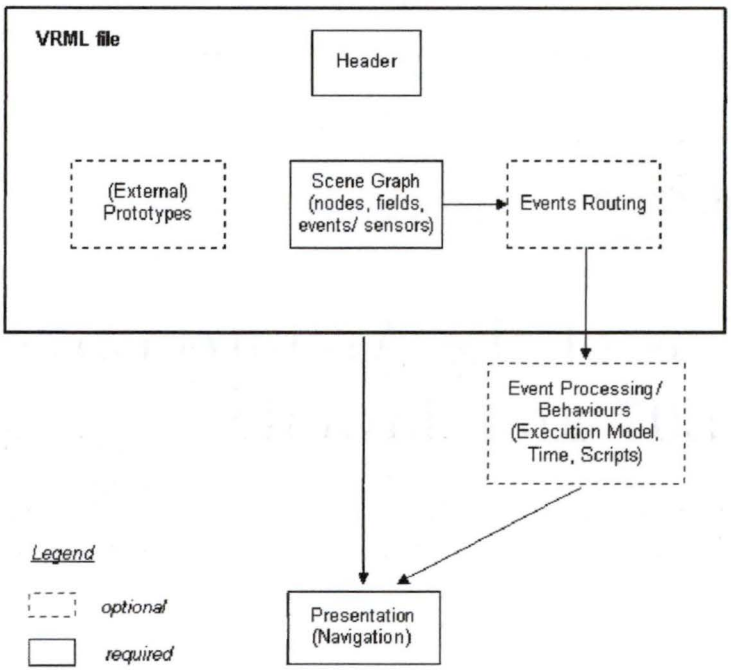


Figure A.1: The complete modelling and visualization mechanism

- may specify hyperlinks to other files and applications
- may define object behaviours.

A.2.2 Scene Graph

A.2.2.1 Description

VRML files describe 3D objects and worlds using a *hierarchical scene graph*, like in Java3D (as discussed in section A.3.2). Entities in the scene graph are called *nodes*[1]. VRML97 defines 54 different node types, including *geometry primitives*, *appearance properties*, *sound* and *sound properties*, and various types of *grouping* nodes. Nodes store their data in *fields*, and VRML97 defines 20 different types of fields that can be used to store everything from a single number to an array of 3D rotations.
The VRML scene graph is a *directed acyclic graph*. Nodes can contain other nodes (some types of nodes may have *"children"*) and may be contained in more than one node (they may have more than one *"parent"*), but a node cannot contain itself. This scene graph structure makes it easy to create large worlds or complicated objects from subparts.

[1]referenced as *objects* in chapter 6

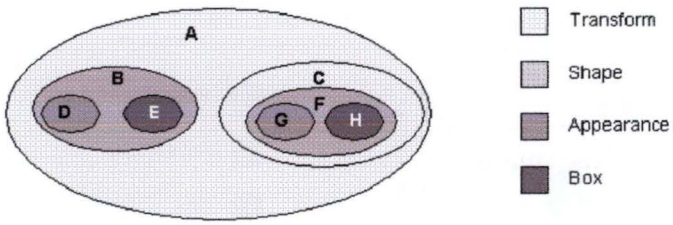


Figure A.2: A hierarchy of *node* components

Therefore, a VRML description consists of a single top-level node (the alter-ego of the *VirtualUniverse* class in Java3D - see section A.3.2). This top-level node may contain other nodes, called *children*, as illustrated in figure A.2. **This figure illustrates the structure of the code in section A.2.2.2.**

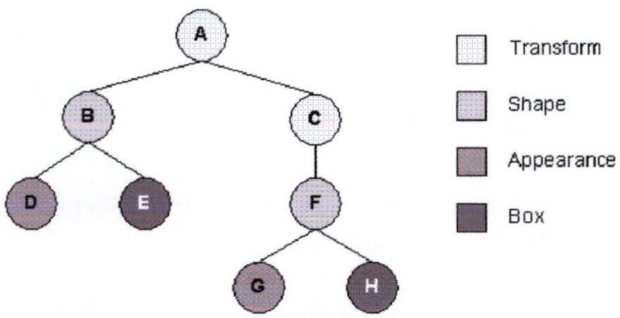


Figure A.3: A node graph

In this case, the top-level node is A, and nodes B and C are its children. The figure can be rearranged to indicate containment, with descending line segments between adjacent nodes. This top-down arrangement is illustrated in figure A.3.

The nodes that make up a scene graph in VRML are **ordered**, as figure A.4 shows it. The dotted line indicates the order in which nodes are processed. Note that the transformations[2] scope is limited by the *Transform* node scope. For instance, *Shape* nodes B and F will be affected by the transformations defined by *Transform* node A while transfomations defined by *Transform* node C will affect only the *Shape* node F.

[2]e.g. translation, rotation and scale

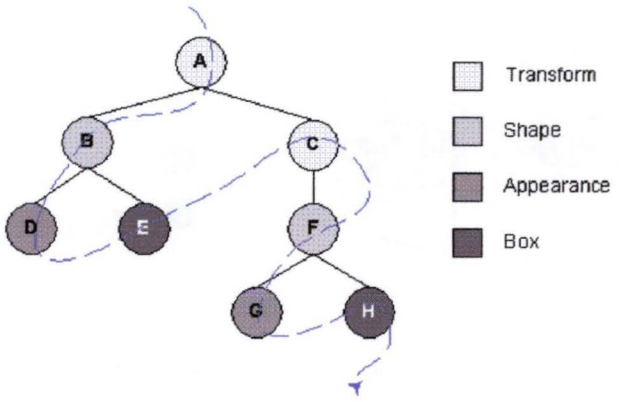


Figure A.4: Traversing a node graph

A node is defined as follows:

Node_Type {Fields_Name Fields_Value}.

Only the node type and the braces are required.

- The node type must be one of a set of predefined types (e.g. *Sphere*, *Box*, *PointLight*, *Anchor*, and so forth).
- Names may be used to identify nodes. They do not have to be unique (although it is always preferable):

  DEF **Node_Name** Node_Type {Fields_Name Fields_Value}

- Fields override the default values of the properties of a node. Each node type defines a set of fields that can be specified explicitly. Fields have names and take values of specific types.

Some node types can have children:

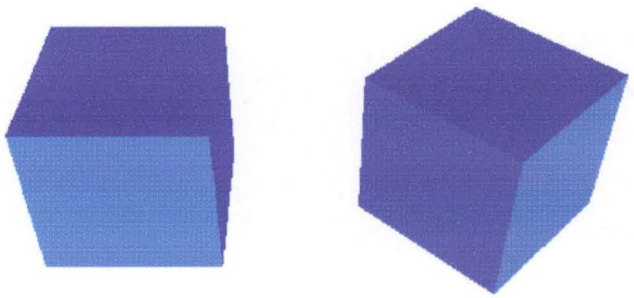
Node_Type {**children**}

For example, the ***Transform*** node includes within its definition the definition of its child nodes.

A.2.2.2 Code example

```
#VRML V2.0 utf8

DEF scene Transform {   # Node A
  translation  -2 0 0              # affect Nodes B and C
  rotation      1 0 0 0.72         # affect Nodes B and C
  children [                       # groups Nodes B and C

    Shape {              # Node B (left-hand cube)
      appearance Appearance {      # Node D
        material Material {
          ambientIntensity  0
          diffuseColor      0 0 0
          specularColor     0 0.92 1
          emissiveColor     0 0 1
          shininess         0.03
          transparency      0
        }}
      geometry Box { }           # Node E
    }# end Shape node

    Transform {          # Node C (right-hand cube)
      translation  4 0 0         # affect Nodes F
      rotation     0 1 0 0.8     # affect Nodes F
      children [
        Shape {              # Node F
          appearance Appearance {     # Node G
            material Material {
              ambientIntensity  0
              diffuseColor      0 0 0
              specularColor     0 0.92 1
              emissiveColor     0 0 1
              shininess         0.03
              transparency      0
```

```
        }}
      geometry Box { }          # Node H
    } # end Shape node (Node F)
  ]} # end Transform node  (Node C)

]} # end  scene Transform node (Node A)
```



The cubes corresponding to the code of section A.2.2.2

A.2.3 Outlines of a virtual universe

This section describes concepts available through a VRML virtual universe.

The interpretation, execution, and presentation of VRML files will typically be undertaken by a mechanism known as a *browser*, which displays the shapes and sounds in the *scene graph*. It performs this by presenting the transformation hierarchy to the user. The transformation hierarchy includes all of the root nodes (the node A in the code example), and root node descendants (the nodes B and C in the code example) that are considered to have one or more particular locations in the *virtual world* (see next paragraph). The transformation hierarchy describes the directly perceptible parts of the virtual world.

This presentation of the transformation hierarchy is known, like in Java3D, as a *virtual universe* (or a *virtual world*) and is navigated in the browser by a human or mechanical entity, known as a *user*. The world is displayed from a particular location; that position and orientation in the world is known as the *viewer*. The browser may define navigation paradigms (such as walking[3] or flying[4]) that enable the user to move the viewer through the virtual world.

In addition to navigation, the browser may provide a mechanism allowing the user to interact with the world through *sensor* nodes, which trigger behaviours. Sensors respond to user interaction with geometric objects in the world, the movement of the user through the world, or the passage of time.

The visual presentation of geometric objects in a VRML world follows a conceptual model designed to resemble the physical characteristics of *light*. The VRML lighting model describes how appearance properties and lights in the world are combined to produce displayed colours.

Figure A.5 illustrates a conceptual model of a VRML browser.

A.3 Java3D architecture

A.3.1 API

Java3D is an object-oriented API. An application based on Java3D API constructs individual graphics elements (as separate objects) and connects them together into a treelike structure called a *scene graph*[5]. The application manipulates these objects using their predefined accessor ("*get*" methods) , mutator ("*set*" methods), and node-linking methods.

This API is a hierarchy of Java classes which serve as the interface to a sophisticated three-dimensional graphics rendering and sound rendering system. The programmer works with high-level constructs (in contrast with low-level constructs like OpenGL) in order to create and manipulate 3D geometric objects.

[3]the user is walking on the **ground** and is stopped by obstacles - see section 5, page 88
[4]the user is flying in the **sky** over obstacles - see section 5, page 88
[5]referenced as a *virtual world* in chapter 6

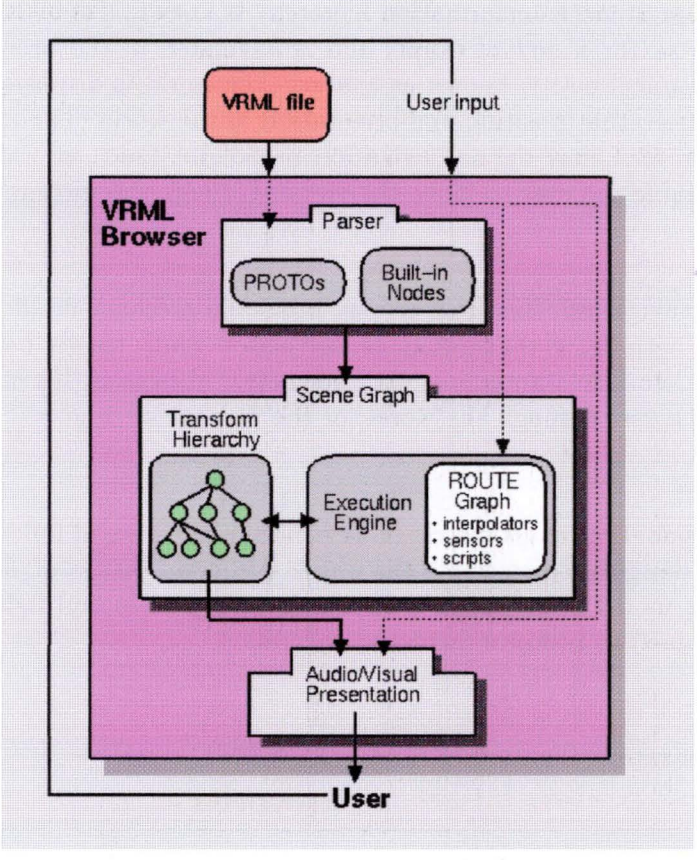


Figure A.5: Conceptual model of a VRML browser

Therefore, any Java3D program is partially assembled from objects from the Java3D class hierarchy. This objects collection describes a *virtual universe*, which is to be rendered. The API defines over 100 classes presented in the **javax.media.j3d** package. These classes are commonly referred to as the Java3D *core* classes. The core class package includes only the lowest-level classes necessary in Java3D programming.

In addition to the Java3D core package, various packages are useful in order to write Java3D programs. Among them stands the **com.sun.j3d.utils** package which is commonly referred to as the Java3D *utility* classes. The utility classes are convenient additions to the core. We shall overview them in few lines. Using utility classes significantly reduces the number of code lines in a Java3D program.

In addition to the Java3D core and utility class packages, any Java3D program uses classes from the **java.awt** package and **javax.vecmath** package.

- The **java.awt** package defines the Abstract Windowing Toolkit (AWT), which creates a window to display the rendering.
- The **javax.vecmath** package defines vector math classes for points, vectors, matrices, and other mathematical objects.

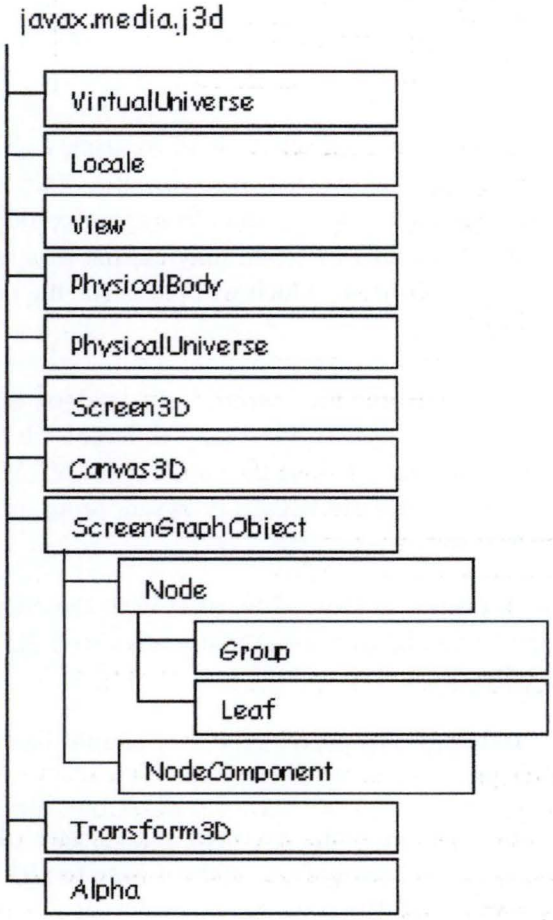


Figure A.6: The first three levels of the Java3D API hierarchy

An overview of the first three levels of the Java3D API hierarchy appears in Figure A.6. These classes are used to build a *scene graph*. A Java3D scene graph is a 3D objects arrangement in a tree structure that completely specifies the *virtual universe* content as well as the rendering process. It will be described in details in the *Scene Graph* section (see section A.3.2).

The *utility* packages

The utility packages bundled with Java3D offer key features. They can be

divided into four major categories:

1. content loaders,
2. scene-graph construction aids,
3. geometry classes, and
4. convenience classes.

- The **content loaders** are designed to load existing content from products like Wavefront and Lightwave[6] into Java3D. Two content loaders are included with Java3D, one for Alias Wavefront object files and one for Lightwave files. A Java3D program may consist in a few lines of code and the content-loader utilities, which will parse the file and dynamically display the content.

  The architecture is extensible for custom-built loaders, and loaders have been developed by third parties for many different file formats. Note that Sun[7] [Sun99a] designed a Java3D renderer for VRML97 scenes. As a matter of fact, various simple VRML browsers programmed in Java3D are available on the web.

- The **scene-graph construction aids** assist with the creation of Java3D scene graphs, trying to abstract away many advanced 3D concepts when they are not needed.

- The **geometry package** has useful geometry shapes like spheres, boxes, cylinders, and cones.

- The **convenience classes** map textures onto scene objects, provide keyboard- or mouse-based navigation, make it easy to pick specific graphics objects in a scene with the mouse, and so forth.

A.3.2 Scene Graph

A Java3D *virtual universe* is created from a *scene graph* which is built using instances of Java3D classes. The scene graph contains a complete description of the entire scene - i.e. the virtual universe - and is assembled from objects to define the *geometry*, *sound*, *light*, *location*, *orientation*, and *appearance* of visual and audio objects. Besides, the scene graph includes viewing information needed to render the scene from a particular point of view. Java3D uses a *directed acyclic graph* (DAG). Figure A.7 shows a typical scene graph.

[6] really famous modelling softwares
[7] http://www.sun.com

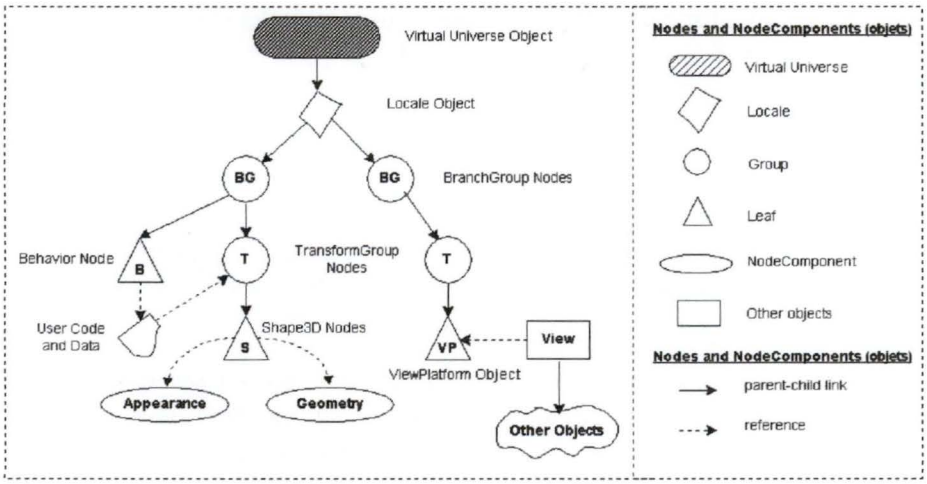


Figure A.7: The Java3D scene graph structure

The *nodes* within the scene graph are instances of Java3D classes. The *arcs* represent both kinds of relationships between the Java3D instances:

- The most common relationship is the *parent-child* relationship. A group node may possess any number of children but **only one** parent, unlike in VRML. A leaf node can have one parent and no children.
- The second relationship is a *reference*. A reference associates a *NodeComponent* object with a *scene graph* Node. NodeComponent objects define the *geometry* and *appearance* attributes used to render the visual objects. Note that the *NodeComponents* and *reference* arcs **are not part** of the scene graph tree. Therefore, several nodes can reference them **without having multi-parents linking**.

The *VirtualUniverse* object provides a grounding for scene graphs. All Java3D scene graphs must connect to a VirtualUniverse object to be displayed.

Below the VirtualUniverse object we find a *Locale* object. The Locale object is simply an anchor to a particular location within the VirtualUniverse. Most Java3D programs need only a single locale centered at (0, 0, 0). All references to location under a Locale in a scene graph are relative to the Locale placement in the VirtualUniverse[8].

The scene graph itself starts with the *BranchGroup* nodes as the root of a subgraph which is called a *branch* graph. Only BranchGroup objects may attach to Locale objects.

[8]In VRML, this *Locale* is implicit !

Two branch graphs stand in figure A.7 and, consequently two BranchGroup nodes: the *Content* graph and the *View* graph.

Attached to the Content graph (the left BranchGroup) we find two subgraphs:

- The first one consists of a user-extended *Behavior leaf* node. The Behavior Leaf node is a powerful, extensible class that allows programmatic interaction with the scene graph at run time.
- The second one consists of a *TransformGroup* node that specifies the position (relative to the Locale), orientation, and scale of the geometric objects in the virtual universe. A single child, a *Shape3D leaf* node, refers to two component objects: a *Geometry* object and an *Appearance* object.
  - the Geometry object describes the geometric shape of a 3D object.
  - the Appearance object describes the appearance of the geometry[9] (colour, texture, material reflection characteristics, and so forth).

The *View* graph (the right BranchGroup) possesses a single subgraph which consists of a *TransformGroup* node and a *ViewPlatform* leaf node. The TransformGroup specifies the position (relative to the Locale), orientation, and scale of the ViewPlatform. **This transformed ViewPlatform object defines the end user's view within the virtual universe.**
Finally, the ViewPlatform is referenced by a *View* object which specifies all of the parameters needed to render the scene from the point of view of the ViewPlatform. Also referenced by the View object are:

- the drawing canvas (*Canvas3D* object) into which Java3D renders
- the screen that contains the canvas
- information about the physical environment

As we may note it, the Scene Graph structure is more complex than in VRML. Appendix B shows an example of Java3D code.

A.3.3 Structure of a Java3D program

We have just overviewed the complex scene graph structure inherent in Java3D. This involves consequences on the program structure. The programmer has to follow some usual steps so as to design a correct program:

[9]The equivalent nodes exist in VRML.

Basic recipe:

1. create a Canvas3D object
2. create a VirtualUniverse object
3. create a Locale object, attaching it to the VirtualUniverse object
4. construct a view branch graph
   (a) create a View object
   (b) create a ViewPlatform object
   (c) create a PhysicalBody object
   (d) create a PhysicalEnvironment object
   (e) attach ViewPlatform, PhysicalBody, PhysicalEnvironment, and Canvas3D objects to View object
5. construct content branch graph(s)
6. compile branch graph(s)
7. insert subgraphs into the Locale

Despite the high level of this API, it is turned out that building simple shapes remains quite complex (**in comparison with VRML**). However, the *SimpleUniverse* class partially solves the problem.

Java3D programs written with the basic recipe often feature View branch graphs with identical structure. That is why the programmer **may** use the *SimpleUniverse* class, where the regularity of view branch graph structure is the key point. *SimpleUniverse* class performs steps 2, 3, and 4 from the basic recipe. Using the SimpleUniverse class in Java3D programming significantly reduces the time and effort needed to create the View branch graph[10]. See figure A.8

Easier recipe:

1. create a Canvas3D object
2. create a SimpleUniverse object which references the earlier Canvas3D object and customize the SimpleUniverse object.
3. construct Content branch graph
4. compile Content branch graph
5. insert content Branch graph into the Locale of the SimpleUniverse

[10]Nevertheless multiple views of the virtual universe are no more possible.

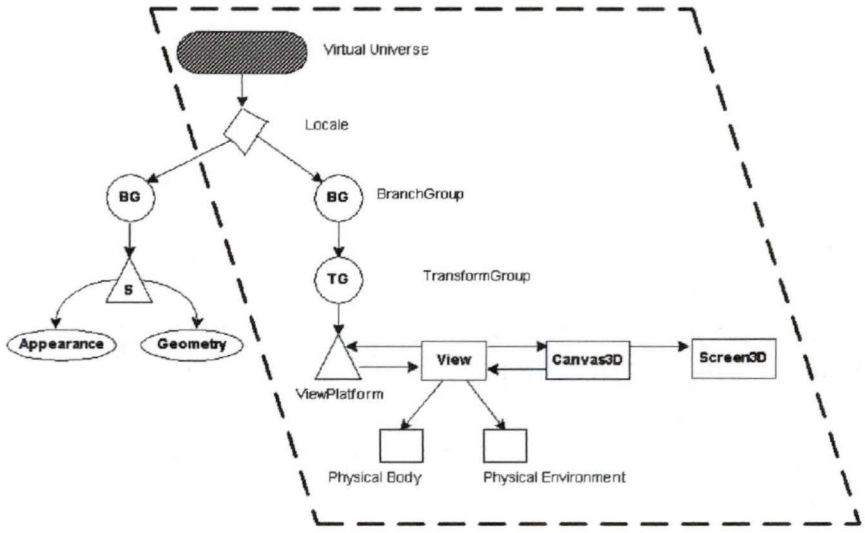


Figure A.8: The dashedlined part is performed by the SimpleUniverse class

A.4 Conclusion

It seems an undisputed fact that the VRML architecture is really more simple and intuitive than the Java3D one. Moreover, despite its high level, the Java3D architecture remains too complex for our simple Business Information Visualization purposes.

According to the simplicity requirement, VRML97 seems the most convenient technological choice.

Appendix B

Comparison Between VRML and Java3D Code

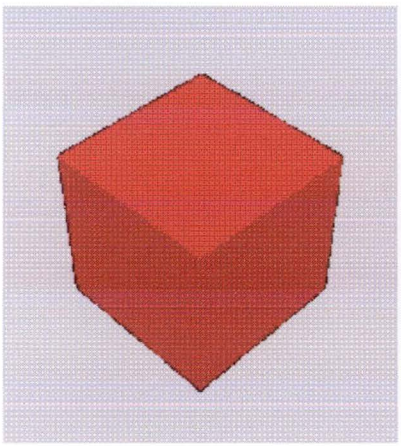
At the University of Port Elizabeth, we have programmed a simple object with each technology.

This is a red cube translated on X and Y axes.

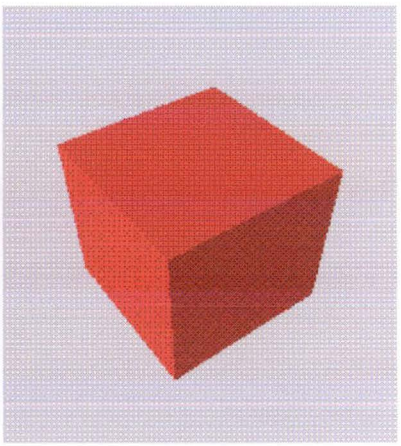
In VRML, we have used the simplest way to build the shape (dealing with the *Box* node).

In Java3D, we have used a more complex way since we have not found any example dealing with the *Box* class. Honestly, it would have taken too much time to search how to implement it.

Therefore, we had rather defined each vertex (24 points) and give the red colour to each of them.



The red cube in VRML97



The red cube in Java3D

B.1 The red cube in VRML97

```
#VRML V2.0 utf8

WorldInfo {
        title "Red Cube"
}

Transform {
  rotation  1 0 0 0.78
  children [
      Transform {
        rotation  0 1 0 0.78
        children [
            Shape {
              appearance Appearance {
                  material Material {diffuseColor 1 0 0 }
              }
              geometry Box {}
            }
        ]
      }
  ]
}
```

B.2 The red cube in Java3D

```
// A simple program in Java3D displaying a rotated red cube

// Java classes
import java.applet.Applet;
import java.awt.BorderLayout;
import java.awt.Frame;
import java.awt.event.*;
```

```
// Java3D classes
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.universe.*;
import javax.media.j3d.*;
import javax.vecmath.*;

public class RedCubeDemo extends Applet
{
  public BranchGroup createSceneGraph()
  {
      // Create the root of the branch graph
      BranchGroup objRoot = new BranchGroup();

      // rotate object has composited transformation matrix
      Transform3D rotate = new Transform3D();
      Transform3D tempRotate = new Transform3D();

      rotate.rotX(Math.PI/4.0d);
      tempRotate.rotY(Math.PI/5.0d);
      rotate.mul(tempRotate);

      TransformGroup objRotate = new TransformGroup(rotate);

      objRoot.addChild(objRotate);
      objRotate.addChild(new RedCube(0.4));

      // Lets Java3D perform optimizations on this scene graph.
      objRoot.compile();

      return objRoot;

  } // end of CreateSceneGraph method
```

```
  // Constructor: Create a simple scene and attach it to the virtual universe
  public RedCubeDemo()
  {
      setLayout(new BorderLayout());
      Canvas3D canvas3D = new Canvas3D(null);
      add("Center", canvas3D);

      BranchGroup scene = createSceneGraph();

      // SimpleUniverse is a Convenience Utility class
      SimpleUniverse simpleU = new SimpleUniverse(canvas3D);

      // This will move the ViewPlatform back a bit so the
      // objects in the scene can be viewed.
      simpleU.getViewingPlatform().setNominalViewingTransform();

      simpleU.addBranchGraph(scene);
  } // end of constructor


  // The following allows this to be run as an application
  // as well as an applet
  public static void main(String[] args)
  {
      Frame frame = new MainFrame(new RedCubeDemo(), 256, 256);
  }
} // end of class RedCubeDemo


/*
 The RED CUBE OBJECT.
 It is designed in defining each vertex (24 points) and a colour
 for each point.
```

We use this technique because is the only one showing how building a simple shape.
We didn't want to spend too much time to find a more "simple" way.
*/

```
class RedCube extends Shape3D
{
  private static final float[] verts = {
   // front face
       1.0f, -1.0f,  1.0f,
       1.0f,  1.0f,  1.0f,
      -1.0f,  1.0f,  1.0f,
      -1.0f, -1.0f,  1.0f,
   // back face
      -1.0f, -1.0f, -1.0f,
      -1.0f,  1.0f, -1.0f,
       1.0f,  1.0f, -1.0f,
       1.0f, -1.0f, -1.0f,
   // right face
       1.0f, -1.0f, -1.0f,
       1.0f,  1.0f, -1.0f,
       1.0f,  1.0f,  1.0f,
       1.0f, -1.0f,  1.0f,
   // left face
      -1.0f, -1.0f,  1.0f,
      -1.0f,  1.0f,  1.0f,
      -1.0f,  1.0f, -1.0f,
      -1.0f, -1.0f, -1.0f,
   // top face
       1.0f,  1.0f,  1.0f,
       1.0f,  1.0f, -1.0f,
      -1.0f,  1.0f, -1.0f,
      -1.0f,  1.0f,  1.0f,
```

```
// bottom face
   -1.0f, -1.0f,  1.0f,
   -1.0f, -1.0f, -1.0f,
    1.0f, -1.0f, -1.0f,
    1.0f, -1.0f,  1.0f,
};

private static final float[] colors = {
// front face
   0.4f, 0.0f, 0.0f,
   0.4f, 0.0f, 0.0f,
   0.4f, 0.0f, 0.0f,
   0.4f, 0.0f, 0.0f,
// back face
   1.0f, 0.0f, 0.0f,
   1.0f, 0.0f, 0.0f,
   1.0f, 0.0f, 0.0f,
   1.0f, 0.0f, 0.0f,
// right face
   1.0f, 0.0f, 0.0f,
   1.0f, 0.0f, 0.0f,
   1.0f, 0.0f, 0.0f,
   1.0f, 0.0f, 0.0f,
// left face
   0.8f, 0.0f, 0.0f,
   0.8f, 0.0f, 0.0f,
   0.8f, 0.0f, 0.0f,
   0.8f, 0.0f, 0.0f,
// top face
   0.7f, 0.0f, 0.0f,
   0.7f, 0.0f, 0.0f,
   0.7f, 0.0f, 0.0f,
   0.7f, 0.0f, 0.0f,
// bottom face
```

```
        1.0f, 0.0f, 0.0f,
        1.0f, 0.0f, 0.0f,
        1.0f, 0.0f, 0.0f,
        1.0f, 0.0f, 0.0f,
    };


    // Constructor
    public RedCube(double scale)
    {
        QuadArray cube = new QuadArray(24, QuadArray.COORDINATES |
                        QuadArray.COLOR_3);

        float scaledVerts[] = new float[verts.length];
        for (int i = 0; i < verts.length; i++)
            scaledVerts[i] = verts[i] * (float)scale;

        cube.setCoordinates(0, scaledVerts);
        cube.setColors(0, colors);

        this.setGeometry(cube);
    }
}
```

Appendix C

A Demo of VRML97 and Java Working Together (Using E.A.I.)

C.1 Aims

The aim of this example made at UPE is to demonstrate the possible interaction between Java and VRML, by using the External Authoring Interface (EAI[1]).

Note : this demo was made at UPE and we used the term "*Bar*" instead of "*Column*" at this moment.

C.2 Description

C.2.1 The VRML97 part

The *3D Bar Chart* was directly built from VRML97 and via the EAI (therefore it is Java that is adding the bars)

- It initially appears with 3 bars of the same height. We have decided that only the height (Y-Scale) of each bar could represent a numeric data. But as we have explained it throughout the thesis, other cues such as colours and transparency could represent data too.
- Two predefined viewpoints ('front' and 'back') and a headlight directional light were designed.

[1] http://vrml.sgi.com/moving-worlds/spec/ExternalInterface.html

  - Viewpoints allow to change the position of the viewer easily. The two predefined positions are in the front and in the back of the chart.
  - The light always illuminates just in front of the viewer (like a light on a miner's helmet).

- The different bars are selectionable by a simple click on it. The interactions via the Java interface will apply only to the selected bar.

Note : in this example, it is impossible to alter the state of the bars by direct interaction on them. For this purpose, the user has to use the Java interface.

C.2.2 The Java part

The interface is created in Java.

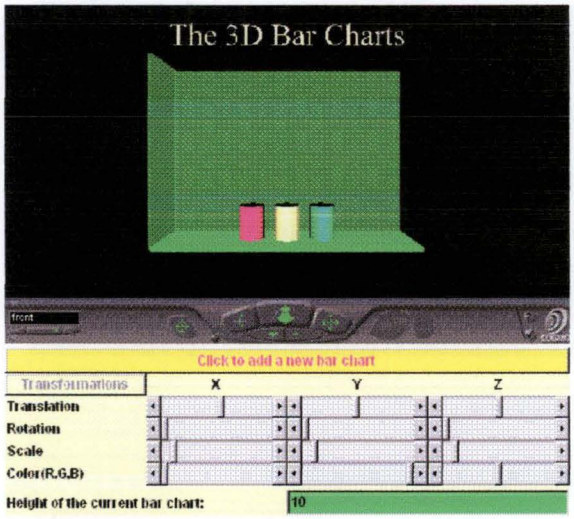
- This interface allows to execute translations and rotations on the three axes of the selected bar as well as to change the scale and the colour. These alterations are performed thanks to scrollbars.

- A *TextField* in the bottom of the window shows the current numeric value of the height (Y-Axis) of the selected bar. The background colour of this TextField always indicates the colour of the selected bar.
  Note: Even if the colour of the bar is changing, the TextField colour is changing at the same time.

- A button labelled "*Click to add a new bar chart*" allows to add a new standard bar. It is a grey bar **always** appearing in the centre of the bar chart. Consequently, if the user adds three bars at a stretch, they appear exactly in the same place.

C.2.3 Screen Shots

ScreenShot_1 : the start screen.

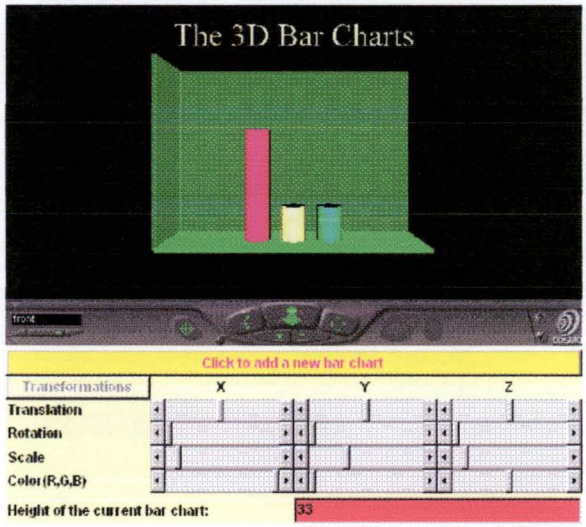


ScreenShot_2 :

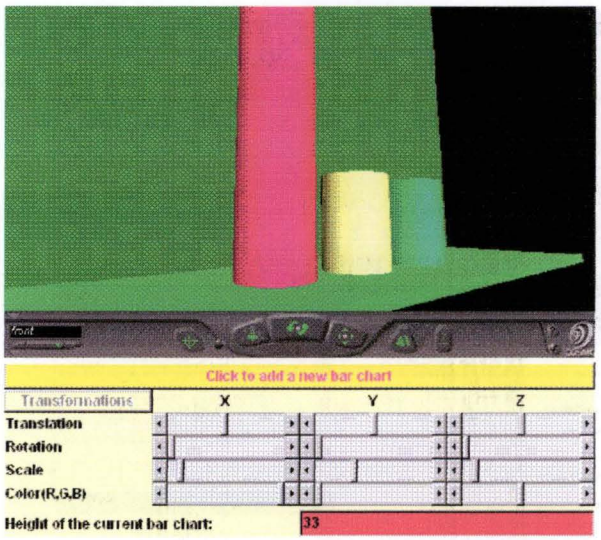
- another bar is selected

- scale on the Y-dimension

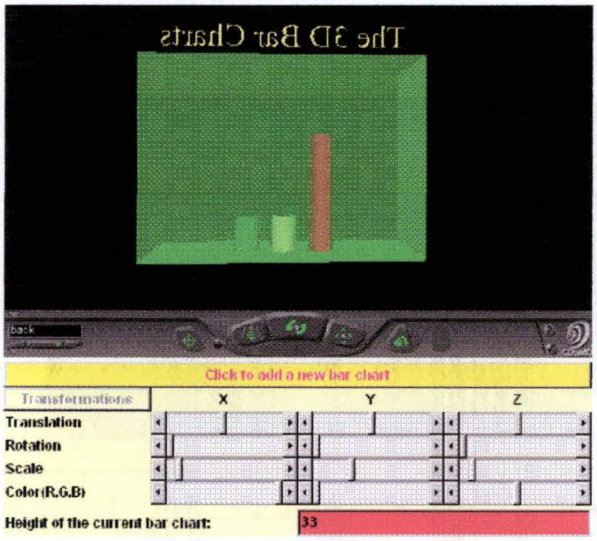
- translation on the Y-axis



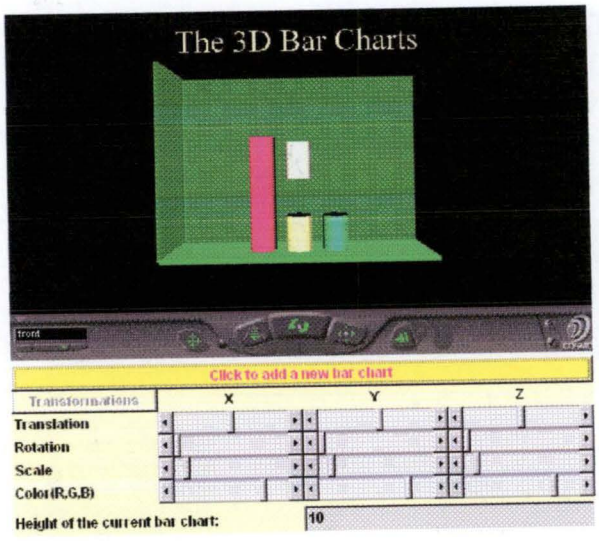
ScreenShot_3 : the viewer has changed his point of view



ScreenShot_4 : the predefined ' back ' view



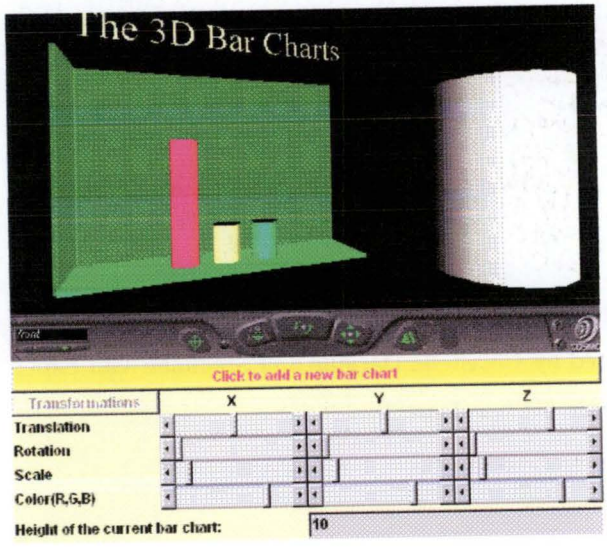
ScreenShot_5 : a new bar is added and translated on the Y-axis



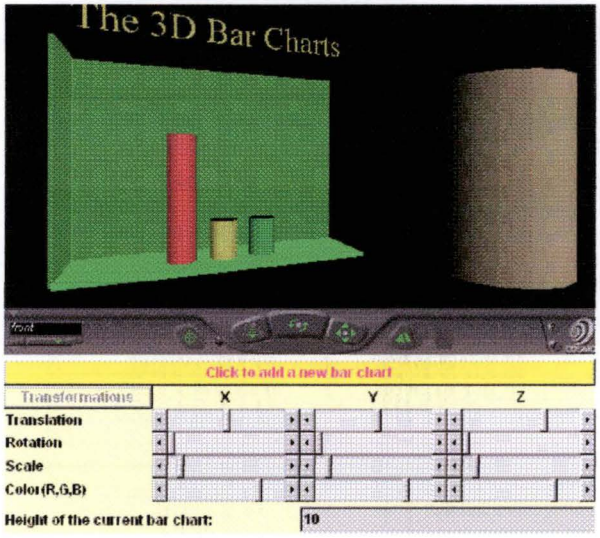
ScreenShot_6 :

- the new bar is translated on the Z-axis

- the viewer has changed his point of view



ScreenShot_7 : the light is turned off



C.2.4 Code

We have programmed by ourselves the VRML file ("trial.wrl") and the "Demo.java" file.

All the other files ("TinyClump.java", "VbRotation.java", "VbVect3f.java") were just reused, except one little modification in "TinyClump.java".

These files were taken from this URL: " http://tecfa.unige.ch/guides/vrml/examples/eai/tiny3D-2"

Here follows the three main code files:

C.2.4.1 Trial.wrl

```
#VRML V2.0 utf8

WorldInfo {
        title "Our first VRML97 World"
        info ["Created by Christelle Darville and Stéphane Van Espen"
            "A simple 3D bar chart"]
       }

# The panels
DEF ROOT Group {
  children[
```

```
Transform {
  translation 0 -1.5 0
  children [
      Shape {
        appearance Appearance {
            material Material {
              diffuseColor 0 0.5 0
              emissiveColor 0 0.8 0
            }
        }
        geometry Box { size 7 0.1 2 }
      } # end Shape
  ]
}

Transform {
  translation -3.5 1 0
  children [
      Shape {
        appearance Appearance {
            material Material {
                  diffuseColor 0 0.5 0
                  emissiveColor 0 0.8 0
                  transparency 0.6
            }
        }
        geometry Box {size 0.1 5 2 }
      } # end Shape
  ]
}

Transform {
  translation 0 1 -1.05
  children [
```

```
            Shape {
              appearance Appearance {
                  material Material {
                        diffuseColor 0 0.5 0
                        emissiveColor 0 0.8 0
                        transparency 0.6
                  }
              }
              geometry Box {size 7 5 0.1 }
            } # end Shape
        ]
    }


# The texts
Transform {
  translation 0 4 0
  children [
      Shape {
        appearance Appearance {
            material Material {
              diffuseColor 1 1 0.5
            }
        }
        geometry Text {
            string "The 3D Bar Charts"
            fontStyle FontStyle {
              size 1
              justify "MIDDLE"
            }
        }
      }
  ]
}
```

```
DEF lumiere DirectionalLight {
  direction 0 0 -1
  color 1 1 0.8
  intensity 0.7
  on TRUE
}

DEF front Viewpoint {
  position 0 1 10
  orientation 0 0 -1 0
  description "front"
}

DEF back Viewpoint {
  position 0 1 -10
  orientation  0 1 0 3.10119
  description "back"
}

]}
```

C.2.4.2 Demo.java

```
// Demo – Simple example of an "authoring tool" in Java
// External Authoring Interface between Java et VRML

// Created by Christelle Darville and Stéphane Van Espen

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
```

```
import vrml.external.field.*;
import vrml.external.exception.*;
import vrml.external.Node;
import vrml.external.Browser;

import TinyClump;
import VbRotation;
import VbVec3f;


public class Demo extends Applet implements ActionListener, AdjustmentListener {
  // One instance variable per slider, to tell what controls what
  Scrollbar transx;
  Scrollbar transy;
  Scrollbar transz;
  Scrollbar rotx;
  Scrollbar roty;
  Scrollbar rotz;
  Scrollbar scalex;
  Scrollbar scaley;
  Scrollbar scalez;
  Scrollbar colr;
  Scrollbar colg;
  Scrollbar colb;

  Label label0;
  Label label1;
  Label label2;
  Label label3;
  Label label4;
  Label label5;
  Label label6;
```

```
Label lab = new Label("Height of the current bar chart:  ");
private TextField tf = new TextField();

Button buttonCylinder, but;
Panel center, sud;

// Browser we're using
Browser browser;

// Root of the scene graph (to which we add our clumps)
Node root;

// Array of the transform hierarchies we've added to the scene
Vector clumps;
int topIndex;

// Current clump we're editing
TinyClump curClump = null;

static int transformRange = 40;


public void init()
{
    setLayout(new BorderLayout());
    setBackground(new Color(1,1,0.8f));
    setFont(new Font("Comic Sans MS", Font.BOLD, 12));

    add(center = new Panel(), BorderLayout.CENTER);
    center.setLayout(new GridLayout(5,4));

    add(sud = new Panel(), BorderLayout.SOUTH);
    sud.setLayout(new GridLayout(1,2));
} // end public void init
```

```
public void start()
{
    tf.setEditable(false);

    // Initialization of our instance variables
    clumps = new Vector();

    topIndex = -1;

    //
    // Initialize connection to Cosmo Player
    //
    browser = Browser.getBrowser(this);

    //
    // Get handle to root of the scene graph
    //
    try
    {
      root = browser.getNode("ROOT");
    }
    catch (InvalidNodeException e)
    {
      System.out.println("PROBLEMS!: " + e);
    }

    // Building the user interface
    super.start();
    center.add(but = new Button("Transformations"));
    but.setEnabled(false);
    but.setBackground(new Color(1,1,0.8f));
    but.setForeground(Color.blue);
```

```
        // Initialize label4
        center.add(label4 = new Label());
        label4.setText("X");
        label4.setAlignment(1);
        // Initialize label5
        center.add(label5 = new Label());
        label5.setText("Y");
        label5.setAlignment(1);
        // Initialize label6
        center.add(label6 = new Label());
        label6.setText("Z");
        label6.setAlignment(1);
        // Initialize label0
        center.add(label0 = new Label());
        label0.setText("Translation");

        // Initialize transx
        center.add(transx = new Scrollbar(Scrollbar.HORIZONTAL, transformRange
/ 2, 1, 0, 40));
        transx.addAdjustmentListener(this);
        // Initialize transy
        center.add(transy = new Scrollbar(Scrollbar.HORIZONTAL, transformRange
/ 2, 1, 0, 40));
        transy.addAdjustmentListener(this);
        // Initialize transz
        center.add(transz = new Scrollbar(Scrollbar.HORIZONTAL, transformRange
/ 2, 1, 0, 40));
        transz.addAdjustmentListener(this);
        // Initialize label1
        center.add(label1 = new Label());
        label1.setText("Rotation");
        // Initialize rotx
        center.add(rotx = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 360));
        rotx.addAdjustmentListener(this);
```

```
// Initialize roty
center.add(roty = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 360));
roty.addAdjustmentListener(this);
// Initialize rotz
center.add(rotz = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 360));
rotz.addAdjustmentListener(this);

// Initialize label2
center.add(label2 = new Label());
label2.setText("Scale");

// Initialize scalex
center.add(scalex = new Scrollbar(Scrollbar.HORIZONTAL, 10, 1, 1, 100));
scalex.addAdjustmentListener(this);
// Initialize scaley
center.add(scaley = new Scrollbar(Scrollbar.HORIZONTAL, 10, 1, 1, 100));
scaley.addAdjustmentListener(this);
// Initialize scalez
center.add(scalez = new Scrollbar(Scrollbar.HORIZONTAL, 10, 1, 1, 100));
scalez.addAdjustmentListener(this);

// Initialize label3
center.add(label3 = new Label());
label3.setText("Color(R,G,B)");
// Initialize colr
center.add(colr = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 255));
colr.addAdjustmentListener(this);
// Initialize colg
center.add(colg = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 255));
colg.addAdjustmentListener(this);
// Initialize colb
center.add(colb= new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 255));
colb.addAdjustmentListener(this);
```

```
    // Initialize buttonCylinder
    add(buttonCylinder = new Button(), BorderLayout.NORTH);
    buttonCylinder.setLabel("Click to add a new bar chart");
    buttonCylinder.setForeground(Color.magenta);
    buttonCylinder.setBackground(Color.yellow);
    buttonCylinder.addActionListener(this);

    // Initialize south part of the Applet
    sud.add(lab);
    sud.add(tf);

    // Adding three shapes
    three_bar();
} // end public void start


public Browser getBrowser()
{
    return browser;
} // end public Browser getBrowser


public void adjustmentValueChanged(AdjustmentEvent TheEvent)
{
    Object TheObject = TheEvent.getSource();
    if (TheObject instanceof Scrollbar)
    {
      if (curClump != null)
      {
        // Depends on which scrollbar changed.
        // TRANSLATION
        if ((TheObject == transx) ||
            (TheObject == transy) ||
            (TheObject == transz))
```

```
{
  float[] val = new float[3];

  // Center about origin
  val[0] = (float) transx.getValue() - (float) transformRange / 2.0f;
  val[1] = (float) transy.getValue() - (float) transformRange / 2.0f;
  val[2] = (float) transz.getValue() - (float) transformRange / 2.0f;
  curClump.set_translation.setValue(val);
} // end if translation

// SCALE
if ((TheObject == scalex) ||
   (TheObject == scaley) ||
   (TheObject == scalez))
{
  float[] val = new float[3];

  // Center about origin
  val[0] = ((float) scalex.getValue()) / 10.0f;
  val[1] = ((float) scaley.getValue()) / 10.0f;
  val[2] = ((float) scalez.getValue()) / 10.0f;
  curClump.set_scale.setValue(val);

  tf.setText(String.valueOf(scaley.getValue()));
} // end if scale

// COLOR
if ((TheObject == colr) ||
   (TheObject == colg) ||
   (TheObject == colb))
{
  float[] val = new float[3];
  // Center about origin
  val[0] = (float) colr.getValue() / 255.0f;
```

```
    val[1] = (float) colg.getValue() / 255.0f;
    val[2] = (float) colb.getValue() / 255.0f;
    curClump.set_diffuseColor.setValue(val);

    tf.setBackground(new Color(val[0],val[1],val[2]));
  } // end if color

  // ROTATION
  if ((TheObject == rotx) ||
      (TheObject == roty) ||
      (TheObject == rotz))
  {
    curClump.xang = rotx.getValue();
    curClump.yang = roty.getValue();
    curClump.zang = rotz.getValue();

    VbRotation xrot =
      new VbRotation(new VbVec3f(1.0f, 0.0f, 0.0f),
                     (float) curClump.xang *
                     (float) (Math.PI / 180.0));
    VbRotation yrot =
      new VbRotation(new VbVec3f(0.0f, 1.0f, 0.0f),
                     (float) curClump.yang *
                     (float) (Math.PI / 180.0));
    VbRotation zrot =
      new VbRotation(new VbVec3f(0.0f, 0.0f, 1.0f),
                     (float) curClump.zang *
                     (float) (Math.PI / 180.0));

    VbRotation r1 = xrot.times(yrot);
    VbRotation r2 = r1.times(zrot);

    VbVec3f axis = new VbVec3f();
    float angle = r2.getValue(axis);
```

```
            float[] val = new float[4];
            float[] axisVal = axis.getValue();
            val[0] = axisVal[0];
            val[1] = axisVal[1];
            val[2] = axisVal[2];
            val[3] = angle;
            curClump.set_rotation.setValue(val);
          } // end if rotation
        } // end if (curClump != null)
    } // end if (TheObject instanceof Scrollbar)
} // end public void adjustmentValueChanged(AdjustmentEvent TheEvent)


public void actionPerformed(ActionEvent TheEvent)
{
  init_shape();
}

// Make a given clump the one we're editing
public void makeCurrent(TinyClump which)
{
  curClump = which;
  initSlidersFromClump();
}

void initSlidersFromClump() {
  float[] val = curClump.scale_changed.getValue();
  scalex.setValue((int) (val[0] * 10.0f ));
  scaley.setValue((int) (val[1] * 10.0f));
  scalez.setValue((int) (val[2] * 10.0f));

  val = curClump.translation_changed.getValue();
  transx.setValue((int) val[0] + (transformRange / 2));
  transy.setValue((int) val[1] + (transformRange / 2));
```

```
    transz.setValue((int) val[2] + (transformRange / 2));

    val = curClump.diffuseColor_changed.getValue();
    colr.setValue((int) (val[0] * 255.0f));
    colg.setValue((int) (val[1] * 255.0f));
    colb.setValue((int) (val[2] * 255.0f));
    float r = val[0];
    float g = val[1];
    float b = val[2];

    rotx.setValue(curClump.xang);
    roty.setValue(curClump.yang);
    rotz.setValue(curClump.zang);

    tf.setText(String.valueOf(scaley.getValue()));
    tf.setBackground(new Color(r,g,b));
  }

  private void init_shape()
  {
    TinyClump newClump = new TinyClump(this, TinyClump.CYLINDER);

    // Add this clump to the end of the vector.
    // To protect the clumps from Java's garbage collector
    clumps.addElement(newClump);

    // Increment the index of the high element
    topIndex++;

    // Initialize slider's values from this clump
    curClump = newClump;
    initSlidersFromClump();

    // Add clump to the scene graph
```

```
    try
    {
    EventInMFNode addChildren = (EventInMFNode) root.getEventIn("addChildren");
      addChildren.setValue(curClump.transArray);
    }
    catch (InvalidEventInException e)
    {
      System.out.println("PROBLEMS!: " + e);
    }
  } // end private void init_shape

  private void three_bar()
  {
    // Middle bar
    init_shape();

    // setting the color
    float[] val = {1f,1f,0.5f};
    curClump.set_diffuseColor.setValue(val);

    // Translating down the bar
    float[] val1 = {0f,-1f,0f};
    curClump.set_translation.setValue(val1);

    // updating the scrollbar and the textfield
    initSlidersFromClump();

    // Left bar
    init_shape();

    // setting the color
    float[] vall = {1f,0f,0.5f};
    curClump.set_diffuseColor.setValue(vall);
```

```
    // Translating the bar
    float[] valg = {-1f,-1f,0f};
    curClump.set_translation.setValue(valg);

    // updating the scrollbar and the textfield
    initSlidersFromClump();

    // Right bar
    init_shape();

    // setting the color
    float[] valr = {0f,1f,0.5f};
    curClump.set_diffuseColor.setValue(valr);

    // Translating the bar
    float[] vald = {1f,-1f,0f};
    curClump.set_translation.setValue(vald);

    // updating the scrollbar and the textfield
    initSlidersFromClump();

  } // end private void three_bar

}
```

C.2.4.3 TinyClump.java

```
// Helper class for encapsulating the objects we add to the Demo scene.

import java.lang.*;
import java.applet.*;
import vrml.external.field.*;
import vrml.external.exception.*;
import vrml.external.Node;
```

```
import vrml.external.Browser;
import Demo;


public class TinyClump extends Object implements EventOutObserver
{
  public final static int CUBE = 0;
  public final static int SPHERE = 1;
  public final static int CONE = 2;
  public final static int CYLINDER = 3;

  // Parent Demo instance
  Demo parent;
  // CosmoPlayer we're using to instantiate these nodes
  Browser browser;

  // Nodes we're interested in
  public Node[] transArray;
  Node transform;
  Node material;

  // EventIns we'll modify
  public EventInSFRotation set_rotation;
  public EventInSFVec3f set_scale;
  public EventInSFVec3f set_translation;
  public EventInSFColor set_diffuseColor;

  // EventOuts we'll query
  public EventOutSFRotation rotation_changed;
  public EventOutSFVec3f scale_changed;
  public EventOutSFVec3f translation_changed;
  public EventOutSFColor diffuseColor_changed;
  public EventOutSFTime touchTime_changed;
```

```
// State associated with the clump
int xang;
int yang;
int zang;

TinyClump(Demo myParent, int whichType) throws IllegalArgumentException
{
  parent = myParent;
  browser = parent.getBrowser();
  xang = yang = zang = 0;

  if ((whichType != TinyClump.CUBE) &&
      (whichType != TinyClump.SPHERE) &&
      (whichType != TinyClump.CONE) &&
      (whichType != TinyClump.CYLINDER))
  {
    throw(new IllegalArgumentException());
  }

  try
  {
    // Instantiate common nodes
    transArray = browser.createVrmlFromString("Transform {}");
    Node[] shapeArray = browser.createVrmlFromString("Shape {}");
    Node[] matArray = browser.createVrmlFromString("Material {}");
    Node[] appArray = browser.createVrmlFromString("Appearance {}");
    Node[] sensArray = browser.createVrmlFromString("TouchSensor {}");
    Node[] geomArray = null;

    if (whichType == TinyClump.CUBE)
    {
      geomArray = browser.createVrmlFromString("Box {}");
    }
```

```
if (whichType == TinyClump.SPHERE)
{
  geomArray = browser.createVrmlFromString("Sphere {}");
}

if (whichType == TinyClump.CONE)
{
  geomArray = browser.createVrmlFromString("Cone {}");
}

if (whichType == TinyClump.CYLINDER)
{
  geomArray = browser.createVrmlFromString("Cylinder {radius 0.35
                                            height 1
                                            bottom FALSE
                                    }");
}

// Put together the hierarchy
transform = transArray[0];
material = matArray[0];
EventInSFNode nodeIn =
  (EventInSFNode) shapeArray[0].getEventIn("appearance");
nodeIn.setValue(appArray[0]);
nodeIn = (EventInSFNode) appArray[0].getEventIn("material");
nodeIn.setValue(material);
nodeIn = (EventInSFNode) shapeArray[0].getEventIn("geometry");
nodeIn.setValue(geomArray[0]);
EventInMFNode nodesIn =
  (EventInMFNode) transform.getEventIn("addChildren");
nodesIn.setValue(shapeArray);
nodesIn.setValue(sensArray);

// Extract EventIns and EventOuts of transform and material node
```

```
    set_rotation = (EventInSFRotation) transform.getEventIn("rotation");
    set_scale = (EventInSFVec3f) transform.getEventIn("scale");
    set_translation = (EventInSFVec3f) transform.getEventIn("translation");
    set_diffuseColor = (EventInSFColor) material.getEventIn("diffuseColor");

  rotation_changed = (EventOutSFRotation) transform.getEventOut("rotation");
    scale_changed = (EventOutSFVec3f) transform.getEventOut("scale");
    translation_changed =
      (EventOutSFVec3f) transform.getEventOut("translation");
    diffuseColor_changed =
      (EventOutSFColor) material.getEventOut("diffuseColor");
    touchTime_changed =
      (EventOutSFTime) sensArray[0].getEventOut("touchTime");

    // Set EventOut callback for this clump's touch sensor,
    // so we can later read in its values when it's clicked
    touchTime_changed.advise(this, null);

  } // end try

  catch (InvalidVrmlException e)
  {
    System.out.println("PROBLEMS!: " + e);
  }

  catch (InvalidEventInException e)
  {
    System.out.println("PROBLEMS!: " + e);
  }

  catch (InvalidEventOutException e)
  {
    System.out.println("PROBLEMS!: " + e);
  }
```

```
    } // end TinyClump(Demo myParent, int whichType)

    // We'll take care of handling the callback every time our
    // TouchSensor is clicked.
    public void callback(EventOut event, double time, Object userData)
    {
      // Make ourselves the current clump
      parent.makeCurrent(this);
    }

}
```

Appendix D

The API Architecture - Java Classes

D.1 Introduction

This appendix shows the code of all the classes that we have implemented during our training at the University of Port Elizabeth.

This classes set constitutes only a part of the whole API architecture. Nevertheless, this set was sufficient to prove that the architecture was functioning properly.

During the writing of the thesis, we have brought some minor modifications to this architecture, notably resulting in the renaming of some classes. We have explicitly indicated these modifications right next the corresponding class names.

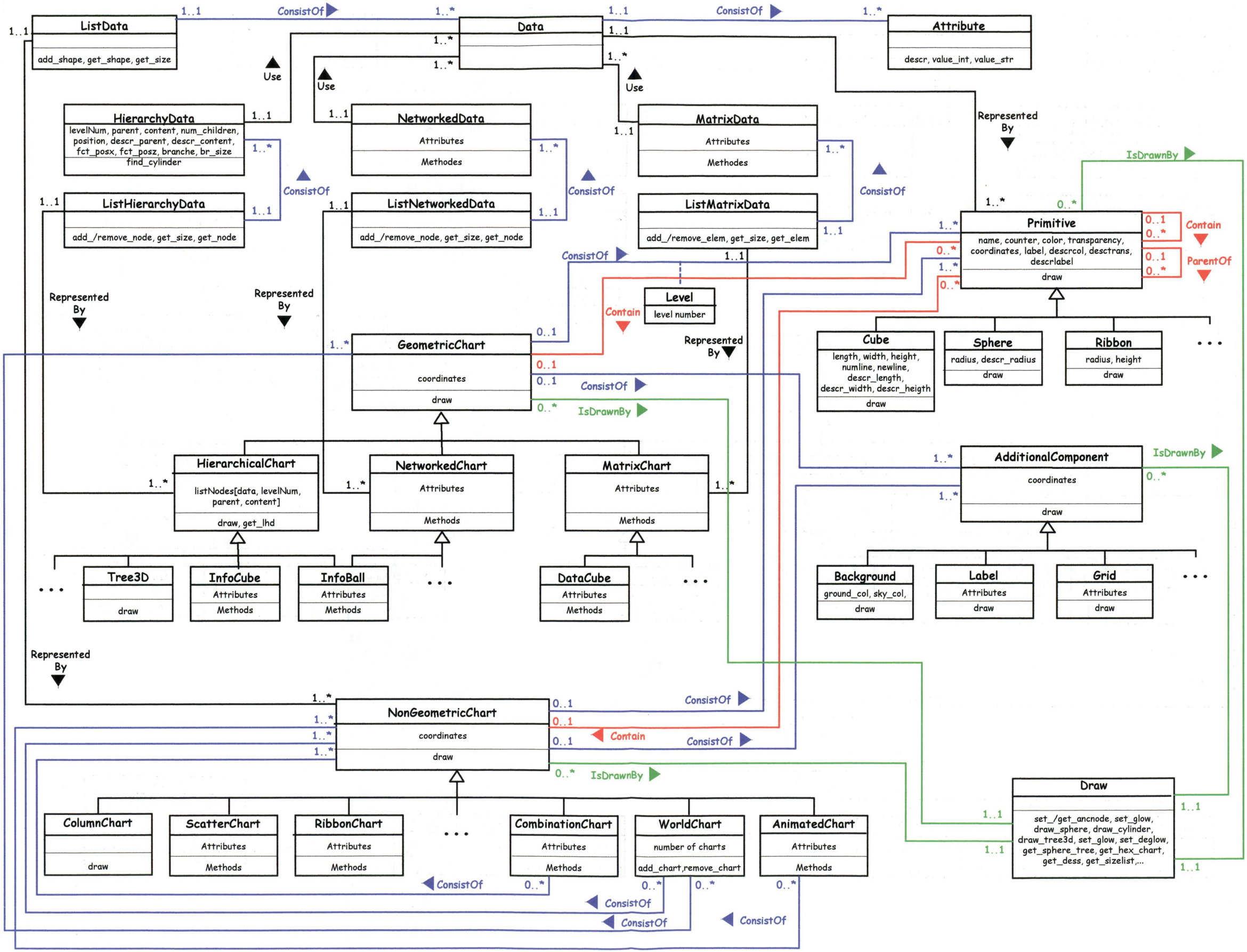
In order to make the code reading easier, an A3 version of the whole API architecture diagram follows the classes code.

D.2 First level

D.2.1 Attribute.java

```
public class Attribute extends Object
{
  // A value can be either a int or a string
```

```
int valint;
String valstr;

// description of the value
String descr;

public Attribute()
{
    valint = 0;
    valstr = "";
    descr = "";
}

public void set_valint(int val)
{
    valint = val;
}

public int get_valint()
{
    return valint;
}

public void set_valstr(String str)
{
    valstr = str;
}

public String get_valstr()
{
    return valstr;
}

public void set_descr(String lab)
```


ListData
add_shape, get_shape, get_size
Data
Attribute
descr, value_int, value_str
ConsistOf
Use
HierarchyData
levelNum, parent, content, num_children, position, descr_parent, descr_content, fct_posx, fct_posz, branche, br_size
find_cylinder
NetworkedData
Attributes
Methodes
MatrixData
Attributes
Methodes
ListHierarchyData
add_/remove_node, get_size, get_node
ListNetworkedData
add_/remove_node, get_size, get_node
ListMatrixData
add_/remove_elem, get_size, get_elem
Represented By
IsDrawnBy
Primitive
name, counter, color, transparency, coordinates, label, descrcol, desctrans, descrlabel
draw
Contain
ParentOf
Level
level number
GeometricChart
coordinates
draw
Cube
length, width, height, numline, newline, descr_length, descr_width, descr_heigth
draw
Sphere
radius, descr_radius
draw
Ribbon
radius, height
draw
HierarchicalChart
listNodes[data, levelNum, parent, content]
draw, get_lhd
NetworkedChart
Attributes
Methods
MatrixChart
Attributes
Methods
Tree3D
draw
InfoCube
Attributes
Methods
InfoBall
Attributes
Methods
DataCube
Attributes
Methods
AdditionalComponent
coordinates
draw
Background
ground_col, sky_col,
draw
Label
Attributes
draw
Grid
Attributes
draw
NonGeometricChart
coordinates
draw
Draw
set_/get_ancnode, set_glow, draw_sphere, draw_cylinder, draw_tree3d, set_glow, set_deglow, get_sphere_tree, get_hex_chart, get_dess, get_sizelist,...
ColumnChart
draw
ScatterChart
Attributes
Methods
RibbonChart
Attributes
Methods
CombinationChart
Attributes
Methods
WorldChart
number of charts
add_chart,remove_chart
AnimatedChart
Attributes
Methods

```
  {
          descr = lab;
  }

  public String get_descr()
  {
     return descr;
  }

}
```

D.2.2 Data.java

```
public class Data extends Object
{
   Attribute tab[];

   // Tab[0] = color
   // Tab[1] = transparency
   // Tab[2] = label
   // Tab[3] = int value used by the radius or lenght(X)
   // Tab[4] = int value used by the height(Y)
   // Tab[5] = int value used by the width(Z)
   // Tab[6] = name
   // Tab[7] = counter
   // Tab[8] = num line
   // Tab[9] = new line

    public Data()
    {
        tab = new Attribute[10];
        for (int i = 0; i <=9; i++)
        {
          tab[i] = new Attribute();
        }
    }
  }
```

D.2.3 ListData.java

```
import java.util.*;

// This class is a list of all the 3D shapes needed to build a chart.

public class Listdata extends Object
{
  Vector vec;

  public Listdata()
  {
      vec = new Vector(10, 1);
  }

  public void add_shape(Data dat)
  {
      vec.addElement(dat);
  }

  public Data get_shape(int i)
  {
      Data temp = (Data) vec.elementAt(i);
      return temp;
  }

  public int get_size()
  {
      return vec.size();
  }

}
```

D.2.4 HierarchyData.java

```
import Data;
import Sphere;

public class HierarchyData extends Object
{
  Data shape;
  int levelNum;
  HierarchyData parent;
  Sphere content;
  int num_children;
  float position[] = {0.0f,0.0f,0.0f};
  String descr_parent, descr_content;
  float fct_posx = 0.0f;
  float fct_posz = 0.0f;
  String branche;
  float br_size;

  // Contructors

  // the root node
  public HierarchyData(Data dat)
  {
      shape = dat;
      parent = null;
      levelNum = 0;
      num_children = 0;
      fct_posx = 48.0f;
      fct_posz = 48.0f;
  }

  public HierarchyData(Data dat, Sphere cont)
  {
      shape = dat;
```

```
        parent = null;
        levelNum = 0;
        content = cont;
        num_children = 0;
        fct_posx = 48.0f;
        fct_posz = 48.0f;
    }

    // The children node
    public HierarchyData(Data dat,HierarchyData par)
    {
        shape = dat;
        parent = par;

        fct_posx = parent.get_fctposx() / 2.0f;
        fct_posz = parent.get_fctposz() / 2.0f;

        position[0] = parent.get_position()[0];
        position[1] = parent.get_position()[1];
        position[2] = parent.get_position()[2];

        // Num_children is updated by the interface (in method parent()). So, it is
always right.
        // we compute the position of this node in function of the parent position and
the parent's
        // number of children
        if (parent.get_numchildren() == 1)
        {
          position[0] = position[0] - fct_posx;
          position[1] = position[1] - 12.0f;
          position[2] = position[2] + fct_posz;
        }
        else
          if (parent.get_numchildren() == 2)
          {
```

```
            position[0] = position[0] + fct_posx;
            position[1] = position[1] - 12.0f;
            position[2] = position[2] + fct_posz;
        }
        else
            if (parent.get_numchildren() == 3)
            {
              position[1] = position[1] - 12.0f;
              position[2] = position[2] - fct_posz;
            }
            levelNum = par.get_levelNum() + 1;
            num_children = 0;

            // Finding the good cylinder
            find_cylinder();
    }

    public HierarchyData(Data dat,HierarchyData par, Sphere cont)
    {
        shape = dat;
        parent = par;
        fct_posx = parent.get_fctposx() / 2.0f;
        fct_posz = parent.get_fctposz() / 2.0f;

        position[0] = parent.get_position()[0];
        position[1] = parent.get_position()[1];
        position[2] = parent.get_position()[2];

        // Num_children is updated by the interface (in method parent()). So, it is
always right.
        // we compute the position of this node in function of the parent position and
the parent's
        // number of children
        if (parent.get_numchildren() == 1)
        {
```

```
        position[0] = position[0] - fct_posx;
        position[1] = position[1] - 12.0f;
        position[2] = position[2] + fct_posz;
      }
      else
        if (parent.get_numchildren() == 2)
        {
            position[0] = position[0] + fct_posx;
            position[1] = position[1] - 12.0f;
            position[2] = position[2] + fct_posz;
        }
        else
            if (parent.get_numchildren() == 3)
            {
              position[1] = position[1] - 12.0f;
              position[2] = position[2] - fct_posz;
            }
            levelNum = par.get_levelNum() + 1;
            num_children = 0;

            // Finding the good cylinder
            find_cylinder();

            content = cont;
}

public Data get_data()
{
    return shape;
}

public HierarchyData get_parent()
{
    return parent;
```

```
}

public int get_levelNum()
{
    return levelNum;
}

public void set_content(Sphere cont)
{
    content = cont;
}

public Sphere get_content()
{
    return content;
}

public void set_numchildren(int i)
{
    num_children = i;
}

public int get_numchildren()
{
    return num_children;
}

public void set_position(float[] pos)
{
    position = pos;
}

public float[] get_position()
{
```

```
    return position;
}

public void set_descrpar(String par)
{
    descr_parent = par;
}

public String get_descrpar()
{
    return descr_parent;
}

public void set_descrcont(String cont)
{
    descr_content = cont;
}

public String get_descrcont()
{
    return descr_content;
}

public float get_fctposx()
{
    return fct_posx;
}

public float get_fctposz()
{
    return fct_posz;
}

public float get_brsize()
```

```
{
    return br_size;
}

public String get_branche()
{
    return branche;
}

private void find_cylinder()
{
    if ((position[0] == -24.0f) &&
       (position[1] == -12.0f) &&
       (position[2] == 24.0f))
    {branche = "Br_1";br_size = 23.0f;}
    else
    if ((position[0] == 24.0f) &&
       (position[1] == -12.0f) &&
       (position[2] == 24.0f))
    {branche = "Br_2";br_size = 23.0f;}
    else
    if ((position[0] == 0.0f) &&
       (position[1] == -12.0f) &&
       (position[2] == -24.0f))
    {branche = "Br_3";br_size = 16.0f;}
    else
    if ((position[0] == -36.0f) &&
       (position[1] == -24.0f) &&
       (position[2] == 36.0f))
    {branche = "Br_1_1";br_size = 11.0f;}
    else
    if ((position[0] == -12.0f) &&
       (position[1] == -24.0f) &&
       (position[2] == 36.0f))
```

```
{branche = "Br_1_2";br_size = 11.0f;}
else
if ((position[0] == -24.0f) &&
    (position[1] == -24.0f) &&
    (position[2] == 12.0f))
{branche = "Br_1_3";br_size = 7.5f;}
else
if ((position[0] == 12.0f) &&
    (position[1] == -24.0f) &&
    (position[2] == 36.0f))
{branche = "Br_2_1";br_size = 11.0f;}
else
if ((position[0] == 36.0f) &&
    (position[1] == -24.0f) &&
    (position[2] == 36.0f))
{branche = "Br_2_2";br_size = 11.0f;}
else
if ((position[0] == 24.0f) &&
    (position[1] == -24.0f) &&
    (position[2] == 12.0f))
{branche = "Br_2_3";br_size = 7.5f;}
else
if ((position[0] == -12.0f) &&
    (position[1] == -24.0f) &&
    (position[2] == -12.0f))
{branche = "Br_3_1";br_size = 11.0f;}
else
if ((position[0] == 12.0f) &&
    (position[1] == -24.0f) &&
    (position[2] == -12.0f))
{branche = "Br_3_2";br_size = 11.0f;}
else
if ((position[0] == 0.0f) &&
    (position[1] == -24.0f) &&
```

```
    (position[2] == -36.0f))
{branche = "Br_3_3";br_size = 7.5f;}
else
if ((position[0] == -42.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 42.0f))
{branche = "Br_1_1_1";}
else
if ((position[0] == -30.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 42.0f))
{branche = "Br_1_1_2";}
else
if ((position[0] == -36.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 30.0f))
{branche = "Br_1_1_3";}
else
if ((position[0] == -18.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 42.0f))
{branche = "Br_1_2_1";}
else
if ((position[0] == -6.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 42.0f))
{branche = "Br_1_2_2";}
else
if ((position[0] == -12.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 30.0f))
{branche = "Br_1_2_3";}
else
if ((position[0] == -30.0f) &&
```

```
    (position[1] == -36.0f) &&
    (position[2] == 18.0f))
{branche = "Br_1_3_1";}
else
if ((position[0] == -18.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 18.0f))
{branche = "Br_1_3_2";}
else
if ((position[0] == -24.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 6.0f))
{branche = "Br_1_3_3";}
else
if ((position[0] == 6.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 42.0f))
{branche = "Br_2_1_1";}
else
if ((position[0] == 18.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 42.0f))
{branche = "Br_2_1_2";}
else
if ((position[0] == 12.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 30.0f))
{branche = "Br_2_1_3";}
else
if ((position[0] == 30.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 42.0f))
{branche = "Br_2_2_1";}
else
```

```
if ((position[0] == 42.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 42.0f))
{branche = "Br_2_2_2";}
else
if ((position[0] == 36.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 30.0f))
{branche = "Br_2_2_3";}
else
if ((position[0] == 18.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 18.0f))
{branche = "Br_2_3_1";}
else
if ((position[0] == 30.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 18.0f))
{branche = "Br_2_3_2";}
else
if ((position[0] == 24.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == 6.0f))
{branche = "Br_2_3_3";}
else
if ((position[0] == -18.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == -6.0f))
{branche = "Br_3_1_1";}
else
if ((position[0] == -6.0f) &&
    (position[1] == -36.0f) &&
    (position[2] == -6.0f))
{branche = "Br_3_1_2";}
```

```
        else
        if ((position[0] == -12.0f) &&
            (position[1] == -36.0f) &&
            (position[2] == -18.0f))
        {branche = "Br_3_1_3";}
        else
        if ((position[0] == 6.0f) &&
            (position[1] == -36.0f) &&
            (position[2] == -6.0f))
        {branche = "Br_3_2_1";}
        else
        if ((position[0] == 18.0f) &&
            (position[1] == -36.0f) &&
            (position[2] == -6.0f))
        {branche = "Br_3_2_2";}
        else
        if ((position[0] == 12.0f) &&
            (position[1] == -36.0f) &&
            (position[2] == -18.0f))
        {branche = "Br_3_2_3";}
        else
        if ((position[0] == -6.0f) &&
            (position[1] == -36.0f) &&
            (position[2] == -30.0f))
        {branche = "Br_3_3_1";}
        else
        if ((position[0] == 6.0f) &&
            (position[1] == -36.0f) &&
            (position[2] == -30.0f))
        {branche = "Br_3_3_2";}
        else {branche = "Br_3_3_3";}
    }

}
```

D.2.5 ListHierarchyData.java

```
import java.util.*;

// This class is a list of all the hierarchical component (HierarchyData)
//      needed to build a hierarchy.

public class ListHierarchyData  extends Object
{
   Vector vec;

   public ListHierarchyData()
   {
       vec = new Vector(10, 1);
   }

   public void add_node(HierarchyData dat)
   {
       vec.addElement(dat);
   }

   public void remove_node(int i)
   {
       vec.removeElementAt(i);
   }

   public HierarchyData get_node(int i)
   {
       HierarchyData temp = (HierarchyData) vec.elementAt(i);
       return temp;
   }

   public int get_size()
   {
       return vec.size();
   }

}
```

D.3 Second level

D.3.1 Shape3d.java [corresponding to the Primitive class in chapter 7]

```
import java.awt.Color;


import Draw;



public abstract class Shape3d extends Object
{
  // Attributes to represent values
  float colors[]= {1.0f,0.0f,0.0f}; // RED
  float position[]= {0.0f,0.0f,0.0f}; // CENTERED
  float transp;
  String label;
  String name;
  int counter;

  // Description of the values
  String descrcol, descrtransp, descrlabel;


  Color color;


  public Shape3d()
  {
      transp = 0.0f;
      label = "";
      name = "";
      descrcol = "";
      descrtransp = "";
      descrlabel = "";
      counter=0;
  }
```

```
// user enters a string with a combobox converted in a array of float
public void set_color(String col)
{
    if (col == "black") color = Color.black;
    else
    if (col == "blue") color = Color.blue;
    else
    if (col == "cyan") color = Color.cyan;
    else
    if (col == "darkGray") color = Color.darkGray;
    else
    if (col == "gray") color = Color.gray;
    else
    if (col == "green") color = Color.green;
    else
    if (col == "lightGray") color = Color.lightGray;
    else
    if (col == "magenta") color = Color.magenta;
    else
    if (col == "orange") color = Color.orange;
    else
    if (col == "pink") color = Color.pink;
    else
    if (col == "red") color = Color.red;
    else
    if (col == "white") color = Color.white;
    else
    if (col == "yellow") color = Color.yellow;
    colors[0] = ((float) color.getRed()) / 255.0f;
    colors[1] = ((float) color.getGreen()) / 255.0f;
    colors[2] = ((float) color.getBlue()) / 255.0f;
}
```

```
public float[] get_color()
{
    return colors;
}

// user enter a percentage between 0% (tranparent) and 100% (opaque)
public void set_transp(int tr)
{
    transp =  ((float) (100-tr))/100.0f;
}

public float get_transp()
{
    return transp;
}

public void set_label(String lab)
{
    label = lab;
}

public String get_label()
{
    return label;
}

public void set_name(String nam)
{
    name = nam;
}

public String get_name()
{
    return name;
```

```
}

public void set_counter(int coun)
{
    counter = coun;
}

public int get_counter()
{
    return counter;
}

public void set_position(float pos[])
{
    position = pos;
}

public float[] get_position()
{
    return position;
}

public void set_descrcol(String dc)
{
    descrcol = dc;
}

public String get_descrcol()
{
    return descrcol;
}

public void set_descrtransp(String dt)
{
```

```
        descrtransp = dt;
    }

    public String get_descrtransp()
    {
        return descrtransp;
    }

    public void set_descrlabel(String dl)
    {
        descrlabel = dl;
    }

    public String get_descrlabel()
    {
        return descrlabel;
    }

    public abstract void draw(Draw dess);

}
```

D.3.2 Hexahedron.java [corresponding to the Cube class in chapter 7]

```
public class Hexahedron extends Shape3d
{

    //Atttibutes related to the shape
    float length,width,heigth;
    String descr_length,descr_width,descr_heigth;
    int num_line;
    String new_line;

    //Constructor
```

```
public Hexahedron(float len,float hei,float wid, Data dat, float[] pos)
{
    super();
    length = len;
    width = wid;
    heigth = hei;
    num_line = dat.tab[8].get_valint();
    new_line = dat.tab[9].get_valstr();

    set_color(dat.tab[0].get_valstr());
    set_transp(dat.tab[1].get_valint());
    set_label(dat.tab[2].get_valstr());
    set_position(pos);
    set_name(dat.tab[6].get_valstr());
    set_counter(dat.tab[7].get_valint());
}

// Length
public void set_length(float len)
{
    length = len;
}

public float get_length()
{
    return length;
}

public void set_descrlen(String dl)
{
    descr_length = dl;
}

public String get_descrlen()
```

```
{
    return descr_length;
}


// Width
public void set_width(float wid)
{
    width = wid;
}

public float get_width()
{
    return width;
}

public void set_descrwid(String dw)
{
    descr_width = dw;
}

public String get_descrwid()
{
    return descr_width;
}


// Height
public void set_heigth(float hei)
{
    heigth = hei;
}

public float get_heigth()
```

```
{
    return heigth;
}

public void set_descrhei(String dh)
{
    descr_heigth = dh;
}

public String get_descrhei()
{
    return descr_heigth;
}

public String get_newline()
{
    return new_line;
}

public int get_numline()
{
    return num_line;
}


// Draw
public void draw(Draw dessin)
{
    dessin.drawhexa(this);
}

}
```

D.3.3 Sphere.java

```
public class Sphere extends Shape3d
{
  //Atttibutes related to the shape
  float radius;
  String descr_radius;

  public Sphere(float rad, Data dat, float[] pos)
  {
      super();
      radius = rad;

      set_color(dat.tab[0].get_valstr());
      set_transp(dat.tab[1].get_valint());
      set_label(dat.tab[2].get_valstr());
      set_position(pos);
      set_name(dat.tab[6].get_valstr());
      set_counter(dat.tab[7].get_valint());
  }

  public void set_radius(float rad)
  {
      radius = rad;
  }

  public float get_radius()
  {
      return radius;
  }

  public void set_descrrad(String dr)
  {
      descr_radius = dr;
  }
```

```
public String get_descrrad()
{
    return descr_radius;
}

public void draw(Draw dessin)
{
    dessin.drawsphere(this);
}

}
```

D.4 Third level

D.4.1 HierchichalChart.java

```
import ListHierarchyData;
import Viewpoint;
import DirectionalLight;
import Draw;

public abstract class HierarchicalChart extends Object
{
  ListHierarchyData lhd;
  float position[]= {0.0f,0.0f,0.0f}; // CENTERED
  Viewpoint vp[]; // maximum 10 viewpoints
  String title;
  DirectionalLight light;

  public HierarchicalChart(ListHierarchyData listhd, float[] pos)
  {
      lhd = listhd;
      lhd.get_node(0).set_position(pos);
      position = pos;
```

```
    vp = new Viewpoint[10];
    title="";
}

public HierarchicalChart(ListHierarchyData listhd,
  float[] pos, String tit)
{
    lhd = listhd;
    lhd.get_node(0).set_position(pos);
    position = pos;
    vp = new Viewpoint[10];
    title = tit;
}

public HierarchicalChart(ListHierarchyData listhd,
  float[] pos, String tit, DirectionalLight li)
{
    lhd = listhd;
    lhd.get_node(0).set_position(pos);
    position = pos;
    vp = new Viewpoint[10];
    title = tit;
    light = li;
}

public void set_Viewpoint(Viewpoint view, int i)
{
    vp[i] = view;
}

public Viewpoint get_Viewpoint(int i)
{
    return vp[i];
}
```

```
public void set_title(String tit)
{
    title = tit;
}

public String get_title()
{
    return title;
}

public void set_position(float[] pos)
{
    position = pos;
}

public float[] get_position()
{
    return position;
}

public void set_dlight(DirectionalLight li)
{
    light = li;
}

public DirectionalLight get_dlight()
{
    return light;
}

public ListHierarchyData get_lhd()
{
    return lhd;
```

```
    }

    public abstract void draw(Draw dess);

}
```

D.4.2 Tree3D.java

```
public class Tree3d extends HierarchicalChart
{
    public Tree3d(ListHierarchyData listhd, float[] pos)
    {
        super(listhd, pos);
    }

    public Tree3d(ListHierarchyData listhd, float[] pos, String tit)
    {
        super(listhd,pos,tit);
    }

    public Tree3d(ListHierarchyData listhd, float[] pos, String tit, DirectionalLight li)
    {
        super(listhd, pos, tit, li);
    }

    public void draw(Draw dessin)
    {
        dessin.drawtree3d(this);
    }

}
```

D.4.3 LinearChart.java [corresponding to the NonGeometricChart class in chapter 7]

```
import Listdata;
import Viewpoint;
import DirectionalLight;
import Draw;


public abstract class LinearChart extends Object
{
  Listdata ld;
  float position[]= {0.0f,0.0f,0.0f}; // CENTERED
  Viewpoint vp[]; // maximum 10 viewpoints
  String title;
  DirectionalLight light;

  public LinearChart(Listdata listd, float[] pos)
  {
     ld = listd;
     position = pos;
     vp = new Viewpoint[10];
     title="";
  }

  public LinearChart(Listdata listd, float[] pos, String tit)
  {
     ld = listd;
     position = pos;
     vp = new Viewpoint[10];
     title = tit;
  }

  public LinearChart(Listdata listd, float[] pos, String tit, DirectionalLight li)
```

```
{
    ld = listd;
    position = pos;
    vp = new Viewpoint[10];
    title = tit;
    light = li;
}

public void set_Viewpoint(Viewpoint view, int i)
{
    vp[i] = view;
}

public Viewpoint get_Viewpoint(int i)
{
    return vp[i];
}

public void set_title(String tit)
{
    title = tit;
}

public String get_title()
{
    return title;
}

public void set_position(float[] pos)
{
    position = pos;
}

public float[] get_position()
```

```
  {
      return position;
  }

  public void set_dlight(DirectionalLight li)
  {
      light = li;
  }

  public DirectionalLight get_dlight()
  {
      return light;
  }

  public Listdata get_ld()
  {
      return ld;
  }

  public abstract void draw(Draw dess);

}
```

D.4.4 BarChart.java [corresponding to the ColumnChart class in chapter 7]

```
public class BarChart  extends LinearChart
{
  public BarChart(Listdata listd, float[] pos)
  {
      super(listd,pos);
  }

  public BarChart(Listdata listd, float[] pos, String tit)
```

```
  {
      super(listd,pos,tit);
  }

  public BarChart(Listdata listd, float[] pos, String tit, DirectionalLight li)
  {
      super(listd,pos,tit,li);
  }

  public void draw(Draw dessin)
  {
      dessin.drawbarchart(this);
  }

}
```

D.5 Fourth level

D.5.1 Draw.java

```
import vrml.external.*;
import vrml.external.field.*;
import vrml.external.exception.*;

import java.util.*;

// For the Tree3D
import ListHierarchyData;
import HierarchyData;
import Sphere;

// for the BarChart
import Listdata;
import Hexahedron;
```

```
public class Draw implements EventOutObserver, Runnable
{
  Browser browser = null;
  Node ancetre = null;
  EventInMFNode addtoanc = null;

  // Sphere
  EventInMFNode addtotransf = null;

  // Vectors to remember all shapes of a chart
  Vector list_shape;
  Vector list_dess;
  Draw dess;

  // Glow
  Thread glow;
  Node[] time_sensor;
  Node[] color_interpol;
  EventInSFTime set_start=null;
  EventOutSFFloat get_fraction;
  EventInSFFloat set_fraction=null;
  EventOutSFColor get_emissive;
  EventInSFColor set_emissive_gen=null;


  // Hexa
  EventInMFNode addtotransfh = null;
  EventInMFNode addtotransh = null;

  Node[] transftitle_bc;
  Node[] shapetitle_bc;

  EventInMFNode remtotransftitle_bc;
```

```
// Constructors
public Draw(Browser brow, Node anc)
{
    browser = brow;
    ancetre = anc;
    list_shape = new Vector(10,1);
    list_dess = new Vector(10,1);

    // Node for the glow effect
    time_sensor =
      browser.createVrmlFromString("DEF event_ts TimeSensor {}");
    color_interpol = browser.createVrmlFromString("DEF event_ci
      ColorInterpolator {key [ 0, 0.5, 1 ] keyValue [ 0 0 0,1 1 1,0 0 0 ]}");

    // Getting access to the startime field for the glow effect
    set_start = (EventInSFTime) time_sensor[0].getEventIn("startTime");

    // Add the observer to the needed eventout
    get_fraction =
      (EventOutSFFloat) time_sensor[0].getEventOut("fraction_changed");
    get_fraction.advise(this, null);

    // Events of the color interpolator
    set_fraction =
      (EventInSFFloat) color_interpol[0].getEventIn("set_fraction");
    get_emissive =
      (EventOutSFColor) color_interpol[0].getEventOut("value_changed");
    get_emissive.advise(this,null);

    glow = new Thread(this);
    glow.setDaemon(true);
    glow.start();
    glow.suspend();
```

```
}


// The ancestor
public void set_ancnode(Node anc)
{
    ancetre = anc;
}

public Node get_ancnode()
{
    return ancetre;
}


// Simulating routed events
public void set_glow()
{
    glow.resume();
}

public void set_deglow()
{
    glow.suspend();
}

public void run()
{
    while(true)
    {
      try
      {
          set_start.setValue((double) (System.currentTimeMillis()/1000));
          glow.sleep(1200);
```

```
        }
        catch (InterruptedException e) {}
    }
}

public void callback(EventOut value, double timestamp, Object data)
{
    if (value.getType() == 6)
    // ROUTE event1_1.fraction_changed TO event2_2.set_fraction
    {
        EventOutSFFloat valeur = (EventOutSFFloat) value;
        set_fraction.setValue(valeur.getValue());
    }
    else
    if (value.getType() == 4)
    // ROUTE event2_2.value_changed TO mat_0.set_emissiveColor
    {
        EventOutSFColor valeur = (EventOutSFColor) value;
        set_emissive_gen.setValue(valeur.getValue());
    }
}


// Drawsphere
public void drawsphere(Sphere sph)
{
    // 1. Getting the needed nodes
    //The main node of the 3D shape
    Node[] transf = browser.createVrmlFromString("Transform{}");

    // The nodes for the sphere
    Node[] shape  = browser.createVrmlFromString("Shape{}");
    Node[] appear = browser.createVrmlFromString("Appearance{}");
    Node[] material = browser.createVrmlFromString("Material{}");
```

```
Node[] geomsph = browser.createVrmlFromString("Sphere
  { radius "+sph.get_radius()+"}");

// The nodes for the label
Node[] transftext = browser.createVrmlFromString("Transform{}");
Node[] shapetext  = browser.createVrmlFromString("Shape{}");
Node[] text = browser.createVrmlFromString("Text{fontStyle FontStyle
   {justify \"MIDDLE\" family \"COMIC SANS MS\" style \"BOLD\" size
3.0}}");

/*
// Node for the glow effect
 Node[] time_sensor = browser.createVrmlFromString("DEF event_ts Time-
Sensor {}");
Node[] color_interpol = browser.createVrmlFromString("DEF event_ci
  ColorInterpolator {key [ 0, 0.5, 1 ] keyValue [ 0 0 0,1 1 1,0 0 0 ]}");
*/

// 2. Getting the attributes events
// Attributes for the sphere
EventInSFColor set_color =
  (EventInSFColor) material[0].getEventIn("diffuseColor");
set_emissive_gen =
  (EventInSFColor) material[0].getEventIn("emissiveColor");
EventInSFFloat set_transp =
  (EventInSFFloat) material[0].getEventIn("transparency");
EventInSFVec3f set_pos =
  (EventInSFVec3f) transf[0].getEventIn("translation");

// Attribute for the text
EventInSFVec3f set_postext =
  (EventInSFVec3f) transftext[0].getEventIn("translation");
EventInMFString set_label =
  (EventInMFString) text[0].getEventIn("string");
```

```
/*
// Getting access to the startime field for the glow effect
set_start = (EventInSFTime) time_sensor[0].getEventIn("startTime");

// Add the observer to the needed eventout
EventOutSFFloat get_fraction =
  (EventOutSFFloat) time_sensor[0].getEventOut("fraction_changed");
get_fraction.advise(this, null);

// Events of the color interpolator
set_fraction = (EventInSFFloat) color_interpol[0].getEventIn("set_fraction");
EventOutSFColor get_emissive =
  (EventOutSFColor) color_interpol[0].getEventOut("value_changed");
get_emissive.advise(this,null);
*/

// 3. Giving the attributes values
// For the sphere
set_color.setValue(sph.get_color());
set_transp.setValue(sph.get_transp());

set_pos.setValue(sph.get_position());

// For the text
float pt[] = new float[3];
pt[0] = 0.0f;
pt[1] = -6.5f;
pt[2] = 0.0f;
set_postext.setValue(pt);
set_label.set1Value(0,sph.get_label());


// 4. Getting the structure events
// The main node of the whole scene
```

```
addtoanc = (EventInMFNode) ancetre.getEventIn("addChildren");

// the nodes for the sphere
addtotransf =
  (EventInMFNode) transf[0].getEventIn("addChildren");
EventInSFNode addtoshape_app =
  (EventInSFNode) shape[0].getEventIn("appearance");
EventInSFNode addtoshape_geom =
  (EventInSFNode) shape[0].getEventIn("geometry");
EventInSFNode addtoapp =
  (EventInSFNode) appear[0].getEventIn("material");

// the nodes for the text
EventInMFNode addtotransftext =
  (EventInMFNode) transftext[0].getEventIn("addChildren");
EventInSFNode addtoshape_text =
  (EventInSFNode) shapetext[0].getEventIn("geometry");

// 5. Building the structure
// Adding the sphere the node of the 3D shape
addtoapp.setValue(material[0]);
addtoshape_app.setValue(appear[0]);
addtoshape_geom.setValue(geomsph[0]);
addtotransf.set1Value(0,shape[0]);

// Adding the text to the node of the 3D shape
addtoshape_text.setValue(text[0]);
addtotransftext.set1Value(0,shapetext[0]);
addtotransf.set1Value(1,transftext[0]);

// Adding the glow nodes to the node of the 3D shape
addtotransf.set1Value(2,time_sensor[0]);
addtotransf.set1Value(3,color_interpol[0]);
```

```
        // Adding the node of the 3D shape to the main node
        addtoanc.set1Value(0,transf[0]);

    }


    // Tree3D
    public void drawtree3d(Tree3d tree)
    {
        // Adding the title
        addtoanc = (EventInMFNode) ancetre.getEventIn("addChildren");

        Node[] transftitle = browser.createVrmlFromString("Transform{}");
        Node[] shapetitle  = browser.createVrmlFromString("Shape{}");
        Node[] title = browser.createVrmlFromString("Text{fontStyle FontStyle
            {justify \"MIDDLE\" family \"COMIC SANS MS\" style \"BOLD\" size
8.0 }}");

        EventInSFVec3f set_postitle =
          (EventInSFVec3f) transftitle[0].getEventIn("translation");
        EventInMFString set_title = (EventInMFString) title[0].getEventIn("string");

        float ptit[] = new float[3];
        ptit[0] = 0.0f;
        ptit[1] = 10.0f;
        ptit[2] = 0.0f;
        set_postitle.setValue(ptit);
        set_title.set1Value(0,tree.get_title());

        EventInMFNode addtotransftitle =
          (EventInMFNode) transftitle[0].getEventIn("addChildren");
        EventInSFNode addtoshape_title =
          (EventInSFNode) shapetitle[0].getEventIn("geometry");

        addtoshape_title.setValue(title[0]);
```

```
addtotransftitle.set1Value(0,shapetitle[0]);
addtoanc.set1Value(1,transftitle[0]);

// Adding the spheres
ListHierarchyData lhd = tree.get_lhd();

for(int i = 0; i <= (lhd.get_size()-1); i++)
{
  // to allow the glow effect, we have to create an Draw object
  // for each sphere
  dess = new Draw(browser,ancetre);

  HierarchyData hd = lhd.get_node(i);
  Sphere sph = new Sphere(4.0f,hd.get_data(),hd.get_position());
  list_shape.addElement(sph);
  sph.draw(dess);
  list_dess.addElement(dess);

  System.out.print("sphere numero "+ String.valueOf(i) + "  drawn:  ");
  System.out.print(String.valueOf(hd.get_position()[0]) + " ");
  System.out.print(String.valueOf(hd.get_position()[1]) + " ");
  System.out.print(String.valueOf(hd.get_position()[2]) + "\n");

  if (i>=1)
  {
     // Showing the cylinder
     System.out.print(hd.get_branche() + "\n");
     Node cyl = browser.getNode((String) hd.get_branche());
   EventInSFNode add_cyl = (EventInSFNode) cyl.getEventIn("geometry");
     Node[] geom = browser.createVrmlFromString("Cylinder
       { radius 0.4 height  " + hd.get_brsize() +  " }");
     add_cyl.setValue(geom[0]);
  }
}
```

```
}

public Sphere get_sphere_tree(int i)
{
    return (Sphere) list_shape.elementAt(i);
}


// Drawhexa
public void drawhexa(Hexahedron hex)
{
    // 1. Getting the needed nodes
    //The main node of the 3D shape
    Node[] htransf = browser.createVrmlFromString("Transform{}");

    // The nodes for the hexahedron
      Node[] htrans = browser.createVrmlFromString("Transform{center -0.5 -0.5
0.5"
      +" scale " + hex.get_length()+ " "
      + hex.get_heigth()+ " " + hex.get_width()+"}");
    Node[] hshape  = browser.createVrmlFromString("Shape{}");
    Node[] happear = browser.createVrmlFromString("Appearance{}");
    Node[] hmaterial = browser.createVrmlFromString("Material{}");
    Node[] geomhex = browser.createVrmlFromString("Box{ size 1 1 1}");

    // The nodes for the label
      Node[] htransftext = browser.createVrmlFromString("Transform{center -0.5
-0.5 0.5
      rotation 0 0 1 1.57}");
    Node[] hshapetext  = browser.createVrmlFromString("Shape{}");
    Node[] htext = browser.createVrmlFromString("Text{fontStyle FontStyle
      {family \"COMIC SANS MS\" style \"BOLD\" size 2.0}}");


    // 2. Getting the attributes events
```

```
// Attributes for the hexahedron
EventInSFColor set_colorh =
  (EventInSFColor) hmaterial[0].getEventIn("diffuseColor");
set_emissive_gen = (EventInSFColor) hmaterial[0].getEventIn("emissiveColor");
EventInSFFloat set_transph =
  (EventInSFFloat) hmaterial[0].getEventIn("transparency");
EventInSFVec3f set_posh =
  (EventInSFVec3f) htransf[0].getEventIn("translation");

// Attribute for the text
EventInSFVec3f set_postexth =
  (EventInSFVec3f) htransftext[0].getEventIn("translation");
EventInMFString set_labelh = (EventInMFString) htext[0].getEventIn("string");


// 3. Giving the attributes values
// For the hexahedron
set_colorh.setValue(hex.get_color());
set_transph.setValue(hex.get_transp());
set_posh.setValue(hex.get_position());

// For the text
float pth[] = new float[3];
pth[0] = 1.0f+ (hex.get_length()/2.0f);
pth[1] = 1.0f+ (hex.get_heigth());
pth[2] = 0.0f;
set_postexth.setValue(pth);
set_labelh.set1Value(0,hex.get_label());

// 4. Getting the structure events
// The main node of the whole scene
addtoanc = (EventInMFNode) ancetre.getEventIn("addChildren");

// the main Transform of the shape (shape + label)
```

```
addtotransfh = (EventInMFNode) htransf[0].getEventIn("addChildren");

// the nodes for the HEXAHEDRON
addtotransh = (EventInMFNode) htrans[0].getEventIn("addChildren");
EventInSFNode addtoshape_apph =
  (EventInSFNode) hshape[0].getEventIn("appearance");
EventInSFNode addtoshape_geomh =
  (EventInSFNode) hshape[0].getEventIn("geometry");
EventInSFNode addtoapph =
  (EventInSFNode) happear[0].getEventIn("material");

// the nodes for the text
EventInMFNode addtotransftexth =
  (EventInMFNode) htransftext[0].getEventIn("addChildren");
EventInSFNode addtoshape_texth =
  (EventInSFNode) hshapetext[0].getEventIn("geometry");


// 5. Building the structure
// Adding the hexahedron the node of the 3D shape
addtoapph.setValue(hmaterial[0]);
addtoshape_apph.setValue(happear[0]);
addtoshape_geomh.setValue(geomhex[0]);
addtotransh.set1Value(0,hshape[0]);
addtotransfh.set1Value(0,htrans[0]);

// Adding the text to the node of the 3D shape
addtoshape_texth.setValue(htext[0]);
addtotransftexth.set1Value(0,hshapetext[0]);
addtotransfh.set1Value(1,htransftext[0]);

// Adding the node of the 3D shape to the main node
addtoanc.set1Value(0,htransf[0]);
}
```

```
//Drawbarchart
public void drawbarchart(BarChart bar)
{
    addtoanc = (EventInMFNode) ancetre.getEventIn("addChildren");
    float pos[] = new float[3];
    float pos_suiv[] = new float[3];
    float cur_long = 0.0f;
    float max_long = 0.0f;

    // Adding the Hexahedron
    Listdata ld = bar.get_ld();

    for(int i = 0; i <= (ld.get_size()-1); i++)
    {
      // to allow the glow effect, we have to create an Draw object
      // for each sphere
      dess = new Draw(browser,ancetre);

      Data dat = ld.get_shape(i);

      float len = ((float) dat.tab[3].get_valint())/20.0f;
      float hei = ((float) dat.tab[4].get_valint())/20.0f;
      float wid = ((float) dat.tab[5].get_valint())/20.0f;

      // Computing the position of current bar

      if (((i==0) && (ld.get_size() == 1)) || ((i==0) &&
          (ld.get_shape(i+1).tab[9].get_valstr().equals("True"))))
      {
          pos[0] = bar.get_position()[0];
          pos[1] = bar.get_position()[1];
          pos[2] = bar.get_position()[2];
```

```
        pos_suiv[0] = bar.get_position()[0];
        pos_suiv[2] = pos[2] - get_maxwid(ld,i, wid) - 2.0f;

        max_long = pos[0] + len;
    }
    else
    {
        if (i==0)
        {
          pos[0] = bar.get_position()[0];
          pos[1] = bar.get_position()[1];
          pos[2] = bar.get_position()[2];
          pos_suiv[0]= pos[0] + len + 1.5f;
          pos_suiv[1]= pos[1];
          pos_suiv[2]= pos[2];
        }
        else {
              if ((i == (ld.get_size()-1) ||
                ((i != (ld.get_size()-1) &&
                 (ld.get_shape(i+1).tab[9].get_valstr().equals("True"))))))
              {
                    pos[0]=pos_suiv[0];
                    pos[1]=pos_suiv[1];
                    pos[2]=pos_suiv[2];
                    pos_suiv[0] = bar.get_position()[0];
                    pos_suiv[2] = pos[2] - get_maxwid(ld,i, wid) - 2.0f;
                    cur_long = pos[0] + len;
                    if (cur_long > max_long) {max_long = cur_long;}
                    System.out.println(pos[2]);
                    System.out.println(pos_suiv[2]);
                    System.out.println("Max_long= " + max_long);
              }
              else
              {
```

```
                    pos[0]=pos_suiv[0];
                    pos[1]=pos_suiv[1];
                    pos[2]=pos_suiv[2];
                    pos_suiv[0]= pos[0] + len + 1.5f;
                }
            }
        }

        // Creating the new bar
        Hexahedron hex = new Hexahedron(len,hei,wid,dat,pos);
        list_shape.addElement(hex);
        hex.draw(dess);
        list_dess.addElement(dess);

       System.out.print("Hexahedron numero "+ String.valueOf(i) + "  drawn:  ");
        System.out.print(String.valueOf(pos[0]) + " ");
        System.out.print(String.valueOf(pos[1]) + " ");
        System.out.print(String.valueOf(pos[2]) + "\n");
    }// end for


    // Getting access to the panels
    Node pan_trans = browser.getNode("PANELS");
    EventInSFVec3f set_pospan =
      (EventInSFVec3f) pan_trans.getEventIn("translation");

    Node pan1 = browser.getNode("pan1"); // pan1 is the floor
    Node pan2 = browser.getNode("pan2"); // pan2 is the left side
    Node pan3 = browser.getNode("pan3"); // pan3 is the background

    EventInSFVec3f set_scale1 = (EventInSFVec3f) pan1.getEventIn("scale");
    EventInSFVec3f set_scale2 = (EventInSFVec3f) pan2.getEventIn("scale");
    EventInSFVec3f set_scale3 = (EventInSFVec3f) pan3.getEventIn("scale");
```

```
// Computing the right place of the panels
set_pospan.setValue(pos_suiv);

// Computing the right size of the panels
float val_scale[] = new float[3];

val_scale[0] = max_long + 1.5f;
val_scale[1] = 0.1f;
val_scale[2] = bar.get_position()[2] - pos_suiv[2] + 1.5f;
set_scale1.setValue(val_scale);

val_scale[0] = 0.1f;
val_scale[1] = get_maxhei() + 1.5f;
val_scale[2] = bar.get_position()[2] - pos_suiv[2] + 1.5f;
set_scale2.setValue(val_scale);

val_scale[0] = max_long + 1.5f;
val_scale[1] = get_maxhei() + 1.5f;
val_scale[2] = 0.1f;
set_scale3.setValue(val_scale);

// Adding the title+
transftitle_bc = browser.createVrmlFromString("Transform{}");
shapetitle_bc  = browser.createVrmlFromString("Shape{}");
Node[] title_bc = browser.createVrmlFromString("Text{}");
Node[] fontstyle = browser.createVrmlFromString("FontStyle{size "
  + (get_maxhei()/1.5f) +" justify \"MIDDLE\"
  family \"COMIC SANS MS\" style \"BOLD\"}");

EventInSFVec3f set_postitle_bc =
  (EventInSFVec3f) transftitle_bc[0].getEventIn("translation");
EventInMFString set_title_bc =
  (EventInMFString) title_bc[0].getEventIn("string");
EventInSFNode add_fstyle =
```

```
    (EventInSFNode) title_bc[0].getEventIn("fontStyle");

  float ptit_bc[] = new float[3];
  ptit_bc[0] = (max_long + 1.5f)/2.0f;
  ptit_bc[1] = get_maxhei() + 5.0f;
  ptit_bc[2] = - bar.get_position()[2] + pos_suiv[2] - 0.5f;
  set_postitle_bc.setValue(ptit_bc);
  set_title_bc.set1Value(0,bar.get_title());

  EventInMFNode addtotransftitle_bc =
    (EventInMFNode) transftitle_bc[0].getEventIn("addChildren");
  remtotransftitle_bc =
    (EventInMFNode) transftitle_bc[0].getEventIn("removeChildren");
  EventInSFNode addtoshape_title_bc =
    (EventInSFNode) shapetitle_bc[0].getEventIn("geometry");

  add_fstyle.setValue(fontstyle[0]);
  addtoshape_title_bc.setValue(title_bc[0]);
  addtotransftitle_bc.set1Value(0,shapetitle_bc[0]);
  addtoanc.set1Value(1,transftitle_bc[0]);

  // Modifying the viewpoint
  Node front = browser.getNode("front");
  EventInSFVec3f set_posview =
    (EventInSFVec3f) front.getEventIn("position");

  float tab_pos[] = new float[3];
  tab_pos[0] = bar.get_position()[0] + (max_long + 1.5f)/2.0f ;
  tab_pos[1] = bar.get_position()[1] + (get_maxhei()* multy());
  tab_pos[2] = (bar.get_position()[2] - pos_suiv[2] + 1.5f)*3.0f;

  set_posview.setValue(tab_pos);

} // End class drawbarchart
```

```
public Hexahedron get_hex_chart(int i)
{
    return (Hexahedron) list_shape.elementAt(i);
}

public Draw get_dess(int i)
{
    return (Draw) list_dess.elementAt(i);
}

public int get_sizelist()
{
    return list_shape.size();
}

private float multy()
{
    if (get_maxhei() < 2.5f) {return 5.0f;}
    else
    if (get_maxhei() < 7.0f) {return 3.0f;}
    else
    if (get_maxhei() < 100.0f) {return 2.0f;}
    else {return 1.5f;}
}


// return the size of the shape with the biggest width of the i line.
private float get_maxwid(Listdata ld, int i, float wid)
{
    int num = ld.get_shape(i).tab[8].get_valint();
    float max = 0.0f;
```

```
    for(int j=0;j<=(get_sizelist()-1);j++)
    {
      if(num == get_hex_chart(j).get_numline())
      {
          if(max < get_hex_chart(j).get_width())
          {
            max = get_hex_chart(j).get_width();
          }
      }
    }
    if (max < wid) {max = wid;}
    return max;
}

private float get_maxhei()
{
    float max = 0.0f;

    for(int j=0;j<=(get_sizelist()-1);j++)
    {
      if(max < get_hex_chart(j).get_heigth())
      {
          max = get_hex_chart(j).get_heigth();
      }
    }
    return max;
}

public void del_title_bc(boolean first)
{
    if (first == false)
    {
      remtotransftitle_bc.set1Value(0,shapetitle_bc[0]);
    }
}


} // End class DRAW
```